МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕЩЕРЯКОВ В.І., ЛАШИНА К.В.

# ІНФОРМАЦІЙНІ СИСТЕМИ І ТЕХНОЛОГІЇ

КОНСПЕКТ ЛЕКЦІЙ
з дисципліни
для студентів 1 курсу денної форми навчання
Рівень вищої освіти – «БАКАЛАВР»
Спеціальність – «Науки про Землю»

Одеса
Одеський державний екологічний університет
2022

УДК 004 (075)

Авторський знак І

Мещеряков В.І, Лашина К.В.

**І:** Інформаційні системи і технології. Конспект лекцій. Одеса, одеський державний екологічний університет, 2022. 171 с.

ISBN 978-966-188-327-6

Анотація

конспекту лекцій з дисципліни Інформаційні системи і технології для студентів 1 курсу денної форми навчання, рівень вищої освіти «Бакалавр», спеціальність Науки про Землю

Наведено технології перетворення інформації мікропроцесорними системами, апаратні і програмні принципи їх побудови, тенденції розвитку технології обробки даних. Розглянуті особливості програмної обробки, накопичення даних, взаємодії з зовнішніми системами. Дана класифікація інформаційних систем обробки великих масивів даних, використання розподілених сховищ даних, мережевих технологій, а також прикладі побудови інформаційних географічних кадастрових, екологічно небезпечних, метеорологічних систем. УДК 004 (075)+2

*Рекомендовано методичною радою Одеського державного екологічного університету Міністерства освіти і науки України як конспект лекцій (протокол №___ від _____._____._____р.)*

ЗМІСТ

# ВСТУП

*Місце і значення навчальної дисципліни*

Предметом вивчення дисципліни “Інформаційні системи і технології” є технології перетворення даних методами цифрових технологій, а також принципи створення інформаційних систем та їх основних складових.

Вивчення навчальної дисципліни сприяє спроможності формування професійних компетенцій, пов’язаних зі спроможністю розуміти основні проблеми у предметної області, вибирати методи і засоби їх вирішення, аналізу науково-технічної проблеми, аналізу науково-технічної літератури, перспективи використання результатів дослідження, готувати результати досліджень, розробляти рекомендації по практичному використанню одержаних результатів.

Метою вивчення дисципліни “Інформаційні системи і технології” є освоєння основних принципів і технологій мікропроцесорної обробки даних систем обробки даних та інформаційних систем збору первинної інформації про зовнішнє середовище, переробки даних, методів автоматичного прийняття рішень відносно екологічного напряму.

Задачі курсу:

вивчення принципів побудови мікропроцесорних систем обробки даних, технології процесорного управління універсальними обчислювальними системами, узгодження апаратної і програмної складових;

організації інформаційних систем обробки інформації, принципів побудови складових системи для можливості аналізу і побудови простих прикладних інформаційних підсистем.

# I ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПЕРЕТВОРЕННЯ ДАНИХ

## 1 ОСНОВНІ ВИЗНАЧЕННЯ

Інформація - це одна з основних універсальних властивостей предметів, явищ, процесів об'єктивної дійсності людини і створених ним керуючих комп'ютерів, що полягає в спроможності сприймати внутрішній стан і вплив навколишнього середовища та зберігати певний час результати його, перетворювати отримані відомості та передавати результати обробки іншим об'єктам.

У вузькому сенсі інформація – це сукупність відомостей про предмети, явища або процеси, що становлять інтерес і підлягає обробці.

Предметом інформатики є знання.

Об’єктом інформатики є система людина – комп’ютер.

Зіставлення понять інформації та знання, представлене у таблиці у формі білістингу, точніше визначає ці поняття. узагальнити відомості та виділити головні відмінності можна наступним чином: інформація – це потік повідомлень, що передаються по матеріальному каналу для динамічного впливу на систему, людину чи суспільство, знання – це фіксоване повідомлення, яке впливає на систему, суспільство чи людину.

Таблиця 1.1

| Інформація | Знання |
|---|---|
| 1. Будь-яке повідомлення будь-якої природи, потік повідомлень<br>2. Зв'язки між елементами повідомлення не такі істотні<br>3. Статистично визначені повідомлення через їхню різноманітність<br>4. Допускається довільне представлення елементів<br>5. Вивчається для розуміння процесів обробки повідомлень<br>6. Важлива динаміка передачі повідомлень у часі<br>7. Є прямий зв'язок із системою сигналів у системі управління | 1. Відсортоване та зафіксоване повідомлення<br>2. Зв'язки між елементами повідомлення повинні допускати свідомість<br>3. Важлива точність визначення повідомлення<br>4. Допускаються лише точні уявлення для одержувача<br>5. Вивчається для розуміння процесів пізнання повідомлень<br>6. Важливе статичне подання підсумкових повідомлень<br>7. Важлива систематизація повідомлень, що формується системою |

У обчислювальній справі розрізняють три засоби представлення знань: предметне, символьне та графічне. Класифікація за видами знань є інформативною, вона головним чином і розглядається.

Історія пізнання поділяється на етапи:

1. чуттєве пізнання (мова та опис)

2. використання спілкування (мова та правила складання фраз)

3. запам'ятовування та фіксація знання (поняття та лист)

4. поширення знання (книга та інші публікації)

5. узагальнення знання (теорія та методологія)

6. застосування знання (алгоритм та дії у пошуку знань)

7. формування нового знання (система та засоби формування нового розуміння).

Перший ступінь у технології пізнання світу полягає у сприйнятті предметів, явищ або процесів, представлених знаками, для введення їх у пам'ять комп'ютера за допомогою символів. Об'єкти відображаються різноманітними знаками, які зв'язуються в щось єдине, представляються у формі, придатній для передачі людині або машині та призначене для виведення нового знання.

Мова є засобом розуміння як навколишнього світу, а й мови. Осмислення мови дозволяє відкрити нові засоби для ефективного уявлення знань про зовнішній та внутрішній стан предметів, явищ або процесів навколишнього світу. Зв'язок між знаками, що відбиває властивості реальних речей, у мовах сформульована у вигляді граматики цієї мови. Формальна сторона призначення граматики становить в узагальненні мови. Сам зв'язок такого сорту дозволяє реалізувати контроль над правильністю передачі змісту або змісту пізнаваного. Мова спілкування людей зазвичай називають природною. Для спілкування з комп'ютерами розроблено формальні мови як низького, так і високого рівнів.

Третій ступінь у пізнавальній діяльності людини полягає у застосуванні писемності та визначенні знання про поняття. Писемність, що обслуговує природні мови, є основною формою визначення знань, при цьому можлива поява нових форм писемності, але її інформативна сутність залишається незмінною.

Четвертий етап розвитку пізнавальної діяльності людини пов'язані з книгою (засобами тиражування знань) та методами представлення, збору та накопичення знань для ефективного сприйняття людиною та комп'ютером. Дослідження засобів тиражування та представлення знань призвело до

формування методів стиснення (кодування) та узагальнення (формалізації) інформації, що представляє знання.

Коли фактів накопичилася така кількість, що запам'ятовування їх стає неможливим, розроблено методи стислого уявлення про сукупність фактів. І тому використовуються твердження про безліч фактів чи правил отримання уявлення факту з відомих уявлень. Твердження про безліч фактів зазвичай виражаються правилами, законами та закономірностями, що пов'язують окремі факти. Загальна постановка питань узгодження знань пов'язана з поняттям специфікації знань.

Специфікація є засобом опису систем, що відноситься до засобів високого рівня і характеризується такими особливостями і навіть перевагами:

- опис системи дається в поняттях завдання, а не засобами або методами її розв'язання;

- специфікація забезпечує верифікацію та перевірку реалізації, скорочує цикли супроводу та модифікації системи;

- специфікація полегшує побудову системи та розуміння у тому, що треба робити, а чи не як треба робити;

- специфікація забезпечує точність, формальність та однозначність опису системи, наочність та читаність властивостей системи, повноту опису;

- кожна специфікація характеризується власними засобами та мовою, що залежить від складності системи;

- специфікація є проміжним засобом опису системи між ескізом та готовою продукцією;

- специфікація є завданням на розробку системи, угодою між замовником та розробником.

Алгоритмічні знання формалізувалися і оформилися в теорію алгоритмів, яка поставила для інформатики надзвичайно корисні знання способи застосування комп'ютерної техніки для отримання нового знання. Сама теорія алгоритмів отримала джерело нового знання лише за розвитку інформатики і з розвитком дискретних перетворювачів інформації.

Знання характеризується своєю само застосовністю та активністю, оскільки саме за допомогою знання можна знаходити нове знання, знання впливає на важелі пошуку нового знання. Накопичене знання забезпечує можливість аналізу цього знання для вироблення знання про знання (само застосовність знання) та формування нового знання (активність знання). Знання

про знання становить основу синтезу нового знання, а синтез нового знання неможливий без аналізу накопиченого знання.

Сьомий крок у пізнавальній діяльності людини полягає у створенні систем формування нового знання з використанням усіх попередніх кроків.

***Види знань***

Будь-яке знання представимо деяким способом – це закон інформатики. Способи уявлення залежать від етапу його освоєння. Відповідно до цього етапу освоєння знання уявлення знань поділяються на сім видів.

Таблиця 1.2

| **Види знань** | **Основні методи їх представлення** |
|---|---|
| Лінгвосеміотичні | Тексти, рисунки і графіка |
| Семантичні | Граматики мов представлення знань, семантичні мережі |
| Концептуальні | Визначення понять, логічні вирази |
| Фактографічні | Масиви, записи, фрейми і таблиці |
| Теоретичні | Аксіоми і правила логічного виводу |
| Алгоритмічні | Алгоритми і програми |
| Кібернетичні | Графи і структури на графах |

Лінгвосеміотичні знання відносяться до опису предметів, явищ або процесів, їх властивостей та характеристик, вони асоціюються з поняттям знака, його вивчення та колекціонування, є результатами фіксації характеристик спостережень у природі та суспільстві.

Семантичні знання пов'язані з процесами розуміння сенсу (змісту) предметів, явищ та процесів, їх властивостей та характеристик. Сенс речей пізнається найрізноманітнішими методами та прийомами.

Концептуальні знання включають відомості про визначення понять (предметів, явищ і процесів або їх властивостей і характеристик), що становить ще один ступінь у пізнанні речей.

Факти - це основа будь-якого пізнання, твердження про факти (розвиток поняття факту) - це продовження бази пізнання, правила виведення тверджень про факти - це вищий рівень пізнання.

Сукупність фактів виробляє теорію. Формальні теорії визначаються як безлічі формул у деякій мові. Безліч формул визначається (крім мови) набором аксіом (істинних формул) та набором правил виведення істинних формул з аксіом. Кожна справжня формула відбиває знання про речі. Істинна формула є формальним записом факту чи твердження фактами. Сукупність знань,

представлених істинними формулами становить теоретичні знання, які є узагальнення фактів і тверджень про факти.

Алгоритмічні знання є щодо новим видом знань та його основним носієм є алгоритми, які реалізують деякі теоретичні становища, виражені сукупністю обґрунтованих теоретичних правил. Синонімом алгоритмічного знання є процедурні знання.

Активна сторона знання виявляється при побудові систем пошуку чи вироблення нового знання, систем управління та взагалі систем як сукупності взаємозалежних та взаємозалежних елементів. Синтез нового знання є головною мозковою діяльністю людини. Знання всіх видів у їх сукупності використовується для цих цілей. Синтез нового знання можливий лише при побудові певної системи. Окремо взяті знаки, мови, поняття, факти, теорії та у певному сенсі ізольовані від перерахованого алгоритми не можуть забезпечити отримання нового знання. Сукупне використання перерахованих предметів, об'єднаних у систему, може призвести до формування нового знання. Знання про системи формування нового знання називаються кібернетичними.

Закон технології пізнання включає узагальнені технологічні операції пізнання, що вибудовуються у послідовність:

ЗНАК - ЗАСІБ ФОРМУВАННЯ МОВИ

МОВА - ЗАСІБ ФОРМУВАННЯ ПОНЯТТЯ

ПОНЯТТЯ - ЗАСІБ ДЛЯ НАКОПЛЕННЯ ФАКТІВ

ФАКТ - ЗАСІБ ДЛЯ УЗАГАЛЬНЕННЯ ЗНАНЬ В ТЕОРІЮ

ТЕОРІЯ - ЗАСІБ ДЛЯ ФОРМУВАННЯ ЗАКОНІВ І АЛГОРИТМІВ

АЛГОРИТМ – ЗАСІБ ДЛЯ ФОРМУВАННЯ СИСТЕМ

СИСТЕМА - ЗАСІБ ДЛЯ ФОРМУВАННЯ НОВИХ ЗНАКІВ.

Сім'ю фундаментальними типами процесів обробки знань з визначення інформатики є:

1. ***Доступ***: дії, за допомогою яких знання надсилаються, запитуються або обробляються конкретним користувачем.
2. ***Розуміння***: дії, з допомогою яких знання оформляються.
3. ***Визначення***: дії, з допомогою яких знання формуються як початкові чи вихідні на формування тверджень.
4. ***Пошук***: пошук та подання неявних знань у явній формі, що уможливлює збір індивідуальних знань для колективного використання в організації у формі колекції фактів.

5. ***Систематизація***: класифікація та кластеризація знань у базах знань з метою їхнього подальшого цілеспрямованого вилучення, підтримки цілісності даних за рахунок реалізації відповідних процесів вирішення завдань.

6. ***Використання***: застосування знань у роботі, прийняття рішень та можливостей через запити прямого користувача.

7. ***Створення***: дії з отримання та оцінки результатів, що становлять нове значення, одержуване на основі бази знань та запитів користувача.

Можна перефразувати архітектуру управління знаннями для представлення архітектури обробки знань:

1. сприйнятливість,
2. представність,
3. зрозумілість,
4. компетентність,
5. інновації,
6. застосовність,
7. ефективність.

Така архітектура використовується для вироблення власної стратегії обробки знань людиною.

***Властивості інформації***

*Релевантність* – здатність інформації відповідати запитам споживача.

*Повнота* - властивість інформації вичерпано для даного споживача характеризувати процес, що відображається.

*Своєчасність* - здатність інформації відповідати потребам споживача зараз.

*Достовірність* – здатність інформації не мати прихованих помилок.

*Доступність* – властивість інформації, яка характеризує можливість отримання її споживачем.

*Захищеність* - властивість, що характеризує неможливість несанкціонованого використання чи зміни.

*Ергономічність* - властивість, що характеризує зручність форми та обсягу інформації з погляду даного споживача.

*Адекватність* - властивість інформації однозначно відповідати об'єкту, що відображається.

# 2 ОБМІН ІНФОРМАЦІЄЮ, АКСІОМАТИКА

При обміні інформацією ***відправник*** і ***одержувач*** проходять кілька взаємозалежних етапів. *Основне завдання відправника* - скласти повідомлення і використовувати канал для його передачі таким чином, щоб обидві сторони зрозуміли і поділили вихідну ідею. Це складно, так як на кожному етапі зміст повідомлення може бути пошкодженим або повністю втрачено.

У процесі руху інформації відбувається її просування але наступним етапам:

- зародження ідеї;
- кодування і вибір каналу;
- передача;
- декодування;
- зворотній зв'язок.

Розглянемо етапи комунікаційного процесу більш докладно для того, щоб показати, які проблеми можуть виникати в його різних точках.

1. ***Зародження ідеї.*** Обмін інформацією починається з формулювання ідеї або відбору інформації. При цьому відправник вирішує, яку саме ідею або повідомлення варто зробити предметом обміну. Його роль полягає в генерації і кодуванні інформації з подальшою передачею іншим учасникам процесу.

Дуже важливо правильно і ретельно сформулювати свою ідею, з тим, щоб вона стала цікавою і привабливою для одержувача. Важливо пам'ятати, що ідея ще не трансформована в слова і не набула іншої форми, в якій вона послужить обміну інформацією. Відправник вирішив тільки, ***що саме*** він хоче передати.

2. ***Кодування і вибір каналу.*** Перш ніж передати ідею, відправник повинен закодувати її за допомогою символів. Наприклад, він може використовувати в якості символів слова, інтонації і жести (мова тіла). Таке кодування перетворює ідею в ***повідомлення.***

Відправник має також вибрати ***канал, сумісний з типом символів,*** використаних для кодування. До деяких загальновідомих каналів відносяться: передача мови, письмових матеріалів, електронні засоби зв'язку, включаючи комп'ютерні мережі та електронну пошту, відеострічки і відео конференції. Якщо канал непридатний для фізичного втілення символів, передача неможлива. Якщо канал не дуже відповідає ідеї, обмін інформацією буде неефективний.

Слід пам'ятати, що вибір засобу повідомлення не повинен обмежуватися єдиним каналом. Часто бажано використовувати два або більше засоби комунікації в певному поєднанні. У зв'язку з цим процес ускладняються,

оскільки відправнику доводиться встановлювати послідовність використання цих коштів і визначати тимчасові інтервали для передачі інформації. Проте вважається, що одночасне використання засобів обміну усною і письмовою інформацією звичайно ефективніше, ніж обмін тільки письмовій.

3. ***Передача.*** На третьому етапі відправник використовує канал для доставки повідомлення (закодованої ідеї або сукупності ідей) одержувачу. Тут мова йде про фізичну передачу повідомлення, що багато людей помилково і приймають за сам процес комунікації. У той же час передача є лише одним з найважливіших етапів, через які необхідно пройти, щоб донести ідею до іншої особи.

4. ***Декодування.*** Після передачі повідомлення відправником одержувач декодує його. **Декодування** - це переклад символів відправника в думці одержувача. Якщо символи, обрані відправником, мають таке ж значення для одержувача, останній знатиме, що саме мав на увазі відправник, коли сформулювалася його ідея. Якщо реакції на ідею не потрібно, процес обміну інформацією на цьому завершується.

5. ***Зворотній зв'язок.*** Обмін інформацією можна вважати ефективним, якщо одержувач продемонстрував розуміння ідеї через зворотній зв’язок. Наприклад, справив дії, яких чекав від нього відправник.

Незважаючи на зовнішню простоту комунікаційного процесу, він рідко протікає без перешкод. Існує безліч потенційних перешкод, які заважають ефективним комунікаціям. Чинники, які порушують чистоту передачі повідомлень, прийнято називати "шумом" в процесі комунікацій.

**"Шум"** - це будь-який чинник, здатний порушити чіткість передачі послання в будь-який момент процесу комунікації.

Джерела шуму варіюються від складності або неточності мови послання до відмінностей у сприйнятті людей одержують його через яких може змінюватися зміст у процесах кодування і декодування. Наприклад, говорять про шумі, коли повідомлення погано закодовані (написані неясно) або погано декодовані (не зрозумілі), або коли канали комунікації не ефективні (увага одержувача відвернута від повідомлення). Таким чином, шум але своєю суттю є бар'єром в процесі комунікації. Певний шум у процесі комунікації є завжди, тому на кожному етапі процесу обміну інформацією відбувається деяке спотворення сенсу. Якщо рівень шуму досить високий, то може відбуватися помітна втрата сенсу послання або навіть повне блокування інформаційного обміну.

Таким чином, **комунікаційний процес** - це послідовність дій при спілкуванні людей. **Мета комунікаційного процесу** - забезпечення розуміння

інформації, що є предметом обміну. Комунікаційний процес має певні елементи і відбувається поетапно. На кожному з етапів може виникати "шум" (перешкоди в комунікаціях), який здатний істотно знижувати їх ефективність.

Головна мета комунікації - обмін різного роду інформацією. Кожне підприємство пронизане мережею інформаційних каналів, які призначені для її збору, аналізу та систематизації.

Пропускна здатність каналу - це обсяг інформації, який може бути переданий через нього за один комунікативний епізод. В цілому комунікація стає більш ефективною при використанні всього безлічі каналів як письмових, так і усних.

На місткість комунікативних каналів впливають **три** фактори:

- здатність обробляти декілька сигналів одночасно;
- можливість забезпечення швидкої, двосторонньої зворотного зв'язку;
- здатність забезпечувати особистий підхід до комунікацій.

З погляду цих можливостей найкращим засобом є ***особисте спілкування.*** Тільки воно гарантує прямий вплив, передачу множинних інформаційних сигналів, негайну зворотний зв'язок і особистий підхід. По суті, ефективність способу комунікації залежить від того, наскільки він підходить для тієї інформації, яку потрібно передати.

Основоположником кількісного подання інформації є Клод Шеннон, який визначив інформацію як міру невизначеності. У загальних словах теорія інформації дає відповіді на такі питання: 15/09/22

1. скільки бітів необхідно для того, щоб представити джерело інформації (наприклад, англійська мова),

2. яка найбільша швидкість, з якою каналом зв'язку можна надійно передавати цифрові дані,

3. з якою ймовірністю можна передавати інформацію (наприклад, зображення) каналом зв'язку з перешкодами.

Відповіді на такі питання зазвичай даються у вигляді деякої фундаментальної межі, яка встановлена самою природою і ніколи не може бути перевищена, спільно з деякою позитивною теоремою, яка стверджує, що за певних умов до цієї фундаментальної межі можна підійти як завгодно близько.

Якщо розглянути природну людську мову за допомогою буквеного опису, то, очевидно, що таке уявлення має високу надмірність, тобто. слова записуються довше, ніж це необхідно для однозначної інтерпретації.

Одне із завдань теорії інформації полягає у вивченні надмірності джерел інформації, наприклад, англійської мови, і, зокрема, відповісти на два наступні питання:

1. яка фундаментальна межа числа букв або цифр, необхідних для однозначного уявлення джерела,

2. як близько до цієї межі можна підійти практично.

Одна з труднощів, що виникають щодо реальних джерел інформації, таких як англійська мова, полягає в тому, що не цілком ясно, як можна побудувати точну математичну модель англійської мови. Крім того, будь-яка модель буде настільки складною, що зрозуміти її основні ідеї буде нелегко. Тому розглянемо ідеалізовану спрощену модель.

Розвитку інтуїтивного розуміння питання сприятиме розгляд такого простого ідеалізованого джерела інформації. Джерело моделюється послідовністю $S_1, S_2, S_3, \ldots,$, що складається зі статистично незалежних реалізацій троїчної випадкової величини $S$, що приймає значення з алфавіту $\{a, b, c\}$, наприклад, позитивно, негативно заряджені частинки та нейтральні. Величина $S$ визначається відповідно до наступного ймовірнісного закону:

$\Pr\{S = a\} = 0{,}7$

$\Pr\{S = b\} = \Pr\{S = c\} = 0{,}15$

Наше завдання полягає в тому, щоб ефективно закодувати послідовність джерела двійкової послідовності (що складається з нулів та одиниць) таким чином, щоб кодованою послідовністю можна було однозначно відновити початкову послідовність джерела.

Один дуже простий спосіб кодування полягає в наступному:

$a \to 00$

$b \to 10$ .

$c \to 11$

За такого кодування за кожен символ джерела припадає 2 двійкових числа (біта). Досягнення більшої ефективності кодування, тобто. меншого числа бітів на символ можна використовувати відомості про статистику джерела. Оскільки символ $a$ подається протягом 70% всього часу, для символу $a$ доцільно використовувати більш коротку послідовність, ніж для символів b і c. Спробуємо закодувати символи так:

$a \to 0$

$b \to 10$.

$c \to 11$

Зауважимо, що оскільки жодна з кодових послідовностей не є префіксом будь-якої іншої кодової послідовності, будь-який ряд кодових послідовностей розшифровується однозначно. Наприклад, послідовність

010110001010

однозначно розшифровується як
$abcaaabb$.

Середня кількість бітів, необхідна на один символ джерела, дорівнює

$$\bar{L} = \Pr\{S=a\}\cdot 1 + \Pr\{S=b\}\cdot 2 + \Pr\{S=c\}\cdot 2 = 0{,}7\cdot 1 + 0{,}15\cdot 2 + 0{,}15\cdot 2 = 1{,}3,$$

що значно менше 2.

Чи можна досягти більшого. Спробуємо ще раз. Припустимо, що замість кодування кожного символу джерела кодуватимемо пари символів джерела: $aa, ab, ac, ba, bb, bc, ca, cb, cc$. Середня кількість бітів на один символ джерела, необхідне при такому кодуванні, дорівнює 1,1975, що краще порівняно з величиною 1,3.

Чи можна досягти ще більшого. Можна спробувати кодувати трійки, четвірки символів джерела і т.д., проте ми досягнемо лише невеликого підвищення ефективності, оцінюваного величиною $\bar{L}$.

Тому виникає питання, чи ми не підходимо до певної фундаментальної межі. Це справді так, і ця фундаментальна межа є, мабуть, найвідомішою з фундаментальних меж теорії інформації. Виражається він через величину, яка носить назву ***ентропії.***

Нехай $U$ - випадкова величина, що приймає значення з кінцевої множини $v$, елементи якого позначаються буквою $u$. Нехай $P(u) = \Pr\{U = u\}, \quad u \in v$. Тоді ентропія випадкової величини $U$, що позначається через H(U), визначається так:

$$H(U) = -\sum_{u \in v} P(u) \log_2 P(u).$$

Логарифми беруться на основі 2. Ентропія залежить тільки від ймовірностей можливих реалізацій і тому є функціоналом кінцевих векторів ймовірностей.

У наведеному прикладі джерела інформації величина була троїчною випадковою величиною $S$, ймовірності трьох можливих реалізацій якої дорівнює відповідно 0,7 0,15 та 0,15. Отже

$$H(S) = -0{,}7\log(0{,}7) - 0{,}15\log(0{,}15) - 0{,}15\log(0{,}15) = 1{,}18129$$

Важливим окремим випадком є $v = \{0,1\}$ і $P(0) = \lambda, \quad P(1) = 1-\lambda, \quad 0 \le \lambda \le 1$. Тоді

$$H(U) = -\lambda\log\lambda - (1-\lambda)\log(1-\lambda) = h(\lambda).$$

Функція має максимум при $\lambda = 1/2$.

Для статистично незалежних реалізацій випадкової величини ентропія визначається виразом

$$H(S_1,\ldots,S_N)=\sum_{k=1}^{N}H(S_k)=NH(S_k).$$

Отже, ентропія джерела $H_S=H(S)$. Приклад джерела, наведеного вище, де $P(a)=0{,}7 \quad P(b)=0{,}15 \quad P(c)=0{,}15$, ентропія $H_S=1{,}18129$.

Визначимо фундаментальні межі ефективності кодування.

Потрібно закодувати послідовність джерела двійковою послідовністю, за якою можна з кінцевою затримкою однозначно відновити послідовність джерела. Ми прагнемо мінімізувати, тобто. середня кількість бітів на символ джерела, необхідне кодування $\bar{L}$. Відповідь виражається через ентропію джерела. При цьому справедливо:

1. за будь-якого способу кодування $L \geq H_S$,

2. можна вибрати такий досить складний спосіб кодування, що величина $\bar{L}$ буде як завгодно близькою до $H_S$.

Таким чином, ентропія $H_S$ є фундаментальною межею для числа двійкових цифр на символ, необхідних для подання джерела.

***Висновки:***

1. Інформація має кількісну оцінку.

2. Залежність між кількістю інформації та кількістю комбінацій, складених з даного алфавіту, - логарифмічна.

3. Загальноприйнята одиниця кількості інформації - двійкова. Виходить вона при виборі одного з двох рівноймовірних символів та дорівнює 1 біт.

4. Кількість інформації визначається не кількістю ситуацій (хоча, безумовно, залежить від нього), а усуненням невідомості про факт, що передається.

5. Кількість інформації визначається щодо первинного алфавіту і не залежить від способу її передачі та довжини повідомлення у вторинному алфавіті.

6. Обсяг інформації визначається щодо вторинного алфавіту і дорівнює добутку середньої довжини повідомлення у вторинному алфавіті на число повідомлень.

7. Кількість інформації не може бути більшою за обсяг тієї ж інформації в порівнянних одиницях.

8. Інформація є поодинокі повідомлення чи його сукупності, які знімають невизначеність, існуючу до надходження.

9. Ентропію появи окремо взятої літери алфавіту визначити не можна, оскільки ентропія характеризує алфавіт загалом і є математичним очікуванням величини — *log* $p_i$. Тому оперувати поняттям «ентропія» щодо появи кодових

слів на виході джерела повідомлень можна лише в тому сенсі, що невизначеність появи кодового слова на виході джерела повідомлень дорівнює ентропії джерела.

## Аксіоматика інформатики

### Перша аксіома інформатики

В складних системах управління підсистема, що управляє, має ієрархічну структуру. Назвемо спостерігачем головний, верхній елемент підсистеми, що управляє. До кожного елементу управляющей підсистеми від спостерігача йде вихідний інформаційний потік $I_{icx}$.(біт), рівний:

$$I_{icx.} = N*H,$$

де $N$— кількість сигналів (команд, документів, даних, вказівок і т. п.), витікаючих від спостерігача; $H$— ентропія цих сигналів ( 0<= $H$<=1).

### Друга аксіома інформатики

Інформаційна напруженість кожного елементу підсистеми $g_j$, що управляє, визначається інформаційною дією на нього спостерігача (вихідним інформаційним потоком) з врахуванням ентропії даного елементу $H_j$:

$$g_j = I_{icx.} / H_j, j=1, m$$

В змістовному аспекті ентропія будь-якого елементу управляющей підсистеми Hj є показником його здібності до творчості, тобто функціонуванню з врахуванням негативного зворотного зв'язку з об'єктом. Якщо $H_j = 1$, це означає, що аналізований елемент управляючої підсистеми лише приймає і ретранслює команди спостерігача і виробляє своєї інформації, тобто. не здійснює коригувальних впливів на об'єкт управління з урахуванням конкретних умов. Якщо $H_j = 0$ - керуючий елемент здійснює управління об'єктом незалежно від спостерігача, повністю самостійно. При $0<H_j<1$ керуючий елемент як ретранслює командну інформацію, що йде від спостерігача, а й вносить власний творчий внесок у інформаційний потенціал керуючої підсистеми. Наприклад, якщо $H_j = 0{,}5$, то елемент удвічі посилює направлений на нього інформаційний потік.

### Третя аксіома інформатики

Інформаційна напруженість всієї підсистеми, що управляє, дорівнює сумі напруженості всіх її елементів, включаючи і спостерігача:

$$Q=\sum g_j, j=1, m,$$

де $g_j$ - інформаційна напруженість конкретного j-го елемента; m - число елементів керуючої підсистеми.

### Четверта аксіома інформатики

Четверта аксіома інформатики встановлює співвідношення між повним інформаційним потоком $I_{пол.}$, що впливає на об'єкт управління за період його переходу в новий цільовий стан, інформаційною напруженістю $Q$ та енергією

об'єкта управління *E*, що витрачається об'єктом управління на перехід у новий стан:

$$E = Q - I_{пол.}.$$

**П'ята аксіома інформатики**

Робота підсистеми *A* (здійснення фізичної роботи, витрати речовинно-енергетичних ресурсів на здійснення інформаційної роботи) складається з двох частин:

$$A = a + b,$$

де *a* — внутрішня робота підсистеми, що управляє, витрачена на компенсацію її вихідної ентропії; *b* - робота, спрямовану об'єкт, тобто. зусилля підсистеми, що управляє, на її інформаційну віддачу.

**Шоста аксіома інформатики**

Корисна робота підсистеми (*b*), що управляє, повинна відповідати повному інформаційному потоку $I_{пол.}$, за аналізований період часу.

## Аксіоми моделювання

**Аксіома 1.** Модель не існує сама по собі, а виступає в тандемі з деяким об'єктом, який вона заміняє в процесі вивчення або проектування.

**Аксіома 2**. Для матеріальних об'єктів модель вторинна, тобто. з'являється як наслідок вивчення та опису цього об'єкта. Для штучних матеріальних об'єктів модель первинна, оскільки передує появі самого об'єкта.

**Аксіома 3**. Модель завжди простіше об'єкта-оригіналу, оскільки вона відображає лише деякі його властивості. Для повноти дослідження об'єкта необхідно будувати та вивчати ряд моделей, що відображають його поведінку чи властивості з різних сторін та з різним ступенем детальності.

**Аксіома 4.** Модель повинна бути подібна до того об'єкта, який вона замінює. Якщо в експериментах з моделлю в одних і тих же умовах модель поводиться так само, як і об'єкт, що моделюється, або розходження в поведінці з точки зору цілей дослідження може бути визнано невеликим, то модель може бути визнана адекватною оригіналу.

**Аксіома 5.** Побудова моделі – не самоціль. Модель будується для того, щоб можна було проводити експерименти не з самим об'єктом, а з більш зручним для цього його замінником.

# 3 ЦИФРОВІ СИСТЕМИ ОБРОБКИ ДАНИХ

Електронна система обробки даних у випадку може мати вигляд рис. 3.1. Як вхідні та вихідні сигнали при цьому можуть використовуватися аналогові сигнали, одиночні цифрові сигнали, цифрові коди, послідовності цифрових кодів. Усередині системи може здійснюватися зберігання, накопичення сигналів (або інформації), але суть від цього не має.

Якщо система цифрова (а мікропроцесорні системи належать до розряду цифрових), то вхідні аналогові сигнали перетворюються на послідовності кодів вибірок за допомогою аналого-цифрового перетворювачів, а вихідні аналогові сигнали формуються з послідовності кодів вибірки за допомогою цифро аналогового перетворювача. Обробка та зберігання інформації здійснюється у цифровому вигляді.


Рисунок 3.1 Електронна система

Характерна риса традиційної цифрової схеми полягає в тому, що алгоритми обробки та зберігання інформації в ній жорстко пов'язані із схемотехнікою системи. Тобто зміна цих алгоритмів можлива лише шляхом зміни структури системи, заміни електронних вузлів, що входять до системи та/або зв'язків між ними. Наприклад, якщо нам потрібна додаткова операція сумування, необхідно додати в структуру системі зайвий суматор. Або якщо потрібна додаткова функція зберігання коду протягом одного такту, то ми повинні додати до структури ще один регістр.

Природно, що практично неможливо зробити в процесі експлуатації, обов'язково потрібен новий виробничий цикл проектування, виготовлення, налагодження всієї системи. Саме тому традиційна цифрова система часто називається системою на "жорсткій логіці".

Будь-яка система на "жорсткій логіці" обов'язково є спеціалізованою системою, побудовано виключно на одне завдання або (рідше) на кілька близьких, заздалегідь відомих завдань. Це має свої безперечні переваги.

По-перше, спеціалізована система (на відміну від універсальної ніколи не має апаратної надмірності, тобто кожен її елемент обов'язково працює на повну силу (звичайно, якщо ця система грамотно спроектована).

По-друге, саме спеціалізована система може забезпечити максимально високу швидкодію, тому що швидкість виконання алгоритмів обробки інформації визначається в ній лише швидкодією окремих логічних елементів та обраною схемою шляхів проходження інформації. А саме логічні елементи завжди мають максимальну на даний момент швидкодію.

Але в той же час великим недоліком цифрової системи на жорсткій логікі є те, що для кожного нового завдання її треба проектувати і виготовляти заново. Це процес тривалий, дорогий, що вимагає високої кваліфікації виконавців. А якщо вирішуване завдання раптом змінюється, то вся апаратура має бути повністю замінена. А це досить марнотратно.

Шлях подолання цього недоліку досить очевидний: треба побудувати таку систему, яка легко адаптувалася під будь-яке завдання, перебудовуватися з одного алгоритму роботи в інший без зміни апаратури. І задавати той чи інший алгоритм ми тоді могли б шляхом введення в систему певної додаткової інформації, що управляє, програми роботи системи (рис. 3.2). Тоді система стане універсальною або програмованою, не жорсткою, а гнучкою. Саме це забезпечує мікропроцесорна система.

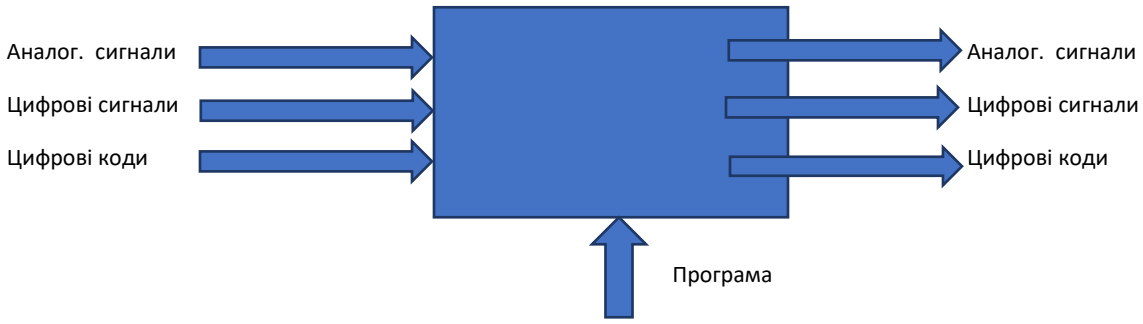



Рисунок 3.2 Програмована (універсальна) електронна система

Але будь-яка універсальність обов'язково призводить до надмірності. Адже розв'язання максимально важкого завдання потребує набагато більше коштів, ніж розв'язання максимально простого завдання. Тому складність універсальної системи має бути такою, щоб забезпечити вирішення найважчого завдання, а при вирішенні простого завдання система працюватиме далеко не на повну силу, використовуватиме не всі свої ресурси. І чим простіше вирішуване завдання, тим більша надмірність, і тим менш виправданою стає універсальність.

Надмірність веде до збільшення вартості системи, зниження її надійності, збільшення споживаної потужності тощо.

Крім того, універсальність, як правило, призводить до суттєвого зниження швидкодії. Оптимізувати універсальну систему так, щоб кожне нове завдання вирішувалося максимально швидко, просто неможливо. Загальне правило таке: чим більша універсальність, гнучкість, тим менша швидкодія. Більш того, для універсальних систем не існує таких завдань (нехай навіть і найпростіших), які вони вирішували б з максимально можливою швидкодією. За все доводиться платити.

Таким чином, можна зробити наступний висновок. Системи на "жорсткій логікі" хороші там, де вирішуване завдання не змінюється тривалий час, де потрібна найвища швидкодія, де алгоритми обробки інформації надто прості. А універсальні, програмовані системи хороші там, де часто змінюються розв'язувані завдання, де висока швидкодія не надто важлива, де складні алгоритми обробки інформації. Тобто будь-яка система хороша на своєму місці.

Ядром будь-якої мікропроцесорної системи є **мікропроцесор** - це той вузол або блок, який робить всю обробку інформації всередині мікропроцесорної системи. Інші вузли виконують лише допоміжні функції: зберігання інформації (зокрема і керуючої інформації, тобто програми), зв'язку із зовнішніми пристроями, зв'язку з користувачем тощо. 22.09.22

Процесор замінює практично всю "жорстку логіку", яка знадобилася б у разі традиційної цифрової системи. Він виконує арифметичні функції (складення, множення тощо), логічні функції (зсув, порівняння, маскування кодів тощо), тимчасове зберігання кодів у внутрішніх регістрах), пересилання кодів між вузлами мікропроцесорної системи та багато іншого. Кількість таких елементарних операцій, що виконуються процесором, може досягати кількох сотень. Процесор можна порівняти з мозком системи.

При цьому треба враховувати, що усі свої операції процесор виконує послідовно, тобто одну за одною, по черзі. Дійсно, існують процесори з паралельним виконанням деяких операцій зустрічаються такі мікропроцесорні системи, в яких декілька процесорів працюють над одною задачкою паралельно, але це виключення.

З іншого боку, послідовне виконання операцій – без сумніву перевага, оскільки дозволяє за допомогою всього одного процесора виконувати любі, найбільш складні алгоритми обробки інформації. Але з іншого боку послідовне виконання операцій призводить до того, що час виконання алгоритму залежить від його складності.

Прості алгоритми виконуються швидше складних. Тобто мікропроцесорна система здатна робити все, але працює не дуже швидко, оскільки усі інформаційні потоки доводиться пропускати через один вузол – мікропроцесор (рис. 3.3). В традиційній цифровій системі можна легко організувати паралельну обробку даних усіх потоків інформації, правда, за рахунок ускладнення системи.

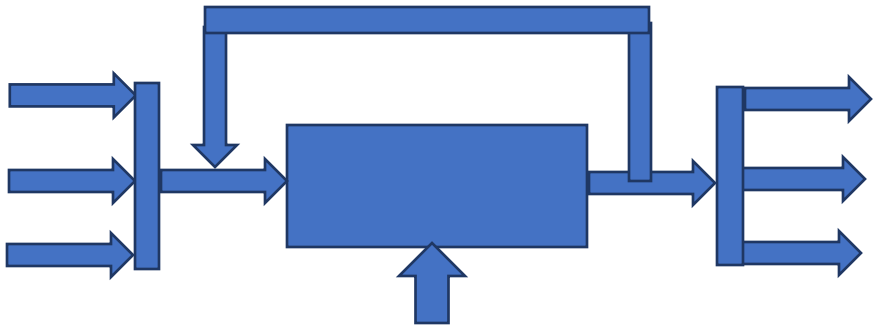


Рисунок 3.3 Інформаційні потоки в мікропроцесорній системі

Отже, мікропроцесор здатний виконувати безліч операцій. Але звідки він дізнається, яку операцію йому потрібно виконувати зараз? Саме це визначається керуючою інформацією, програмною.

Програма є набором **команд (інструкцій**), тобто цифрових кодів, розшифрувавши які, процесор дізнається, що йому треба робити. Програма від початку і до кінця складається людиною, програмістом, а процесор виступає в ролі виконавця цієї програми, жодної ініціативи він не виявляє (якщо, звичайно, справний).

Тому порівняння процесора з мозком не надто коректне. Він лише виконавець того алгоритму, який заздалегідь склав для нього людина. Будь-яке відхилення від цього алгоритму може бути викликане лише несправністю процесора або інших вузлів мікропроцесорної системи.

Усі команди, що виконуються процесором, утворюють систему команд процесора. Структура та обсяг системи команд процесора визначають його швидкодію, гнучкість, зручність використання. Усього команд у процесора може бути від кількох десятків до кількох сотень.

Система команд може бути розрахована на вузьке коло розв'язуваних задач (у спеціалізованих процесорів) або максимально широке коло завдань (у універсальних процесорів). Коди команд можуть мати різну кількість розрядів (займати від одного до кількох байт). Кожна команда має свій час виконання, тому час виконання всієї програми залежить не тільки від кількості команд у програмі, але й від того, які команди використовуються.

Для виконання команд структуру процесора входять внутрішні регістри, арифметико-логічний пристрій АЛУ, мультиплексори, буфери, регістри та інші вузли. Робота всіх вузлів синхронізується загальним зовнішнім тактовим сигналом процесора. Тобто процесор є досить складним цифровим пристроєм (рис. 3.4).


Рисунок 3.4 Структура простого процесора

Для розробника мікропроцесорних систем інформація про тонкощі внутрішньої структури не надто важлива. Розробник повинен розглядати процесор як чорну скриньку, яка у відповідь на вхідні та керуючі коди робить ту чи іншу операцію та видає вихідні сигнали. Розробнику необхідно знати систему команд, режими роботи процесора, а також правила взаємодії процесора із зовнішнім світом, або, як їх ще називають, протоколи обміну інформацією. Про внутрішню структуру процесора треба знати лише те, що потрібно вибору тієї чи іншої команди, тієї чи іншої режиму роботи.

Для досягнення максимальної універсальності та спрощення протоколів обміну інформацією в мікропроцесорних системах застосовується так звана шинна структура зв'язків між окремими пристроями, що входять до системи. Суть шинної структури зв'язку зводиться до такого. При класичній структурі зв'язків (рис. 3.5) всі сигнали та коди між пристроями передаються окремими лініями зв'язку. Кожен пристрій, що входить до системи, передає свої сигнали та коди

незалежно від інших пристроїв. При цьому в системі виходить дуже багато ліній зв'язку та різних протоколів обміну інформацією.

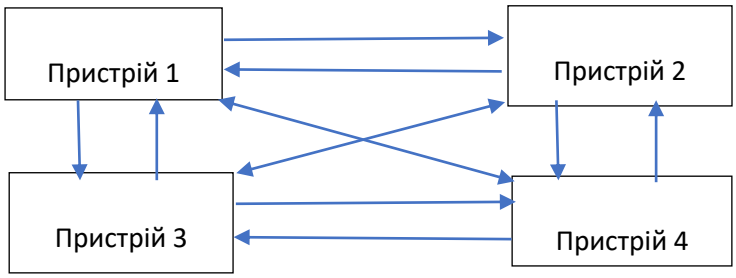



Рисунок 3.5 Класична структура зв’язків

При шинної структурі зв'язків (рис. 3.6) всі сигнали між пристроями передаються по одних і тих самих лініях зв'язку, але в різний час (це називається мультиплексованою передачею). Причому передача по всіх лініях зв'язку може здійснюватися обох напрямках (так звана двонаправлена передача). В результаті кількість ліній зв'язку значно скорочується, а правила обміну (протоколи) спрощуються. Група ліній зв'язку, якими передаються сигнали чи коди називається шиною (англ. Bus).

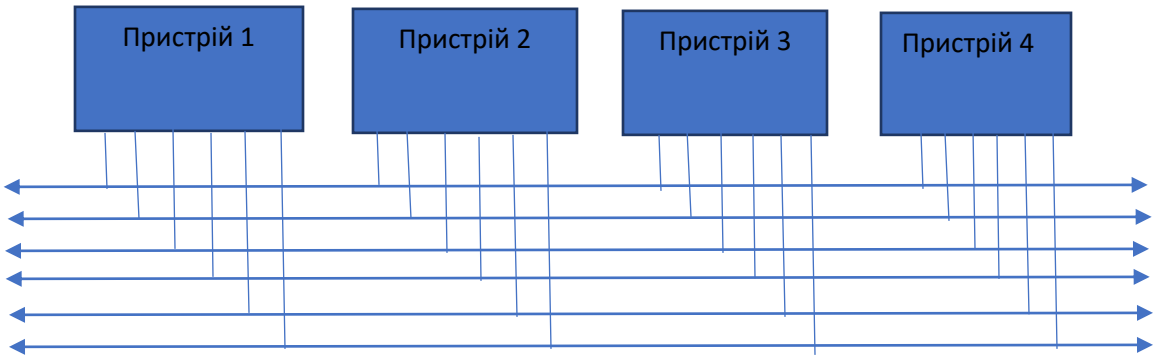



Рисунок 3.6 Шинна структура зв’язків

Зрозуміло, що при шинній структурі зв'язків легко здійснюється пересилання всіх інформаційних потоків у потрібному напрямку, наприклад, їх можна пропустити через процесор, що дуже важливо для мікропроцесорної системи. Однак при шинній структурі зв'язків вся інформація передається лініями зв'язку послідовно в часі, по черзі, що знижує швидкодію системи в порівнянні з класичною структурою зв'язку.

Велика перевага шинної структури зв'язків полягає в тому, що всі пристрої, підключені до шини, повинні приймати та передавати інформацію за тими самими правилами (протоколам обміну інформацією по шині). Відповідно,

всі вузли, відповідальні за обмін із шиною у цих пристроях, мають бути однакові, уніфіковані.

Істотний недолік шинної структури пов'язаний з тим, що всі пристрої підключаються до кожної лінії зв'язку паралельно. Тому будь-яка несправність будь-якого пристрою може вивести з ладу всю систему. Якщо вона псує лінію зв'язку. З цієї ж причини налагодження системи з шинною структурою зв'язків досить складне і зазвичай потребує спеціального обладнання.

Типова структура мікропроцесорної системи приведена на рис. 3.7.

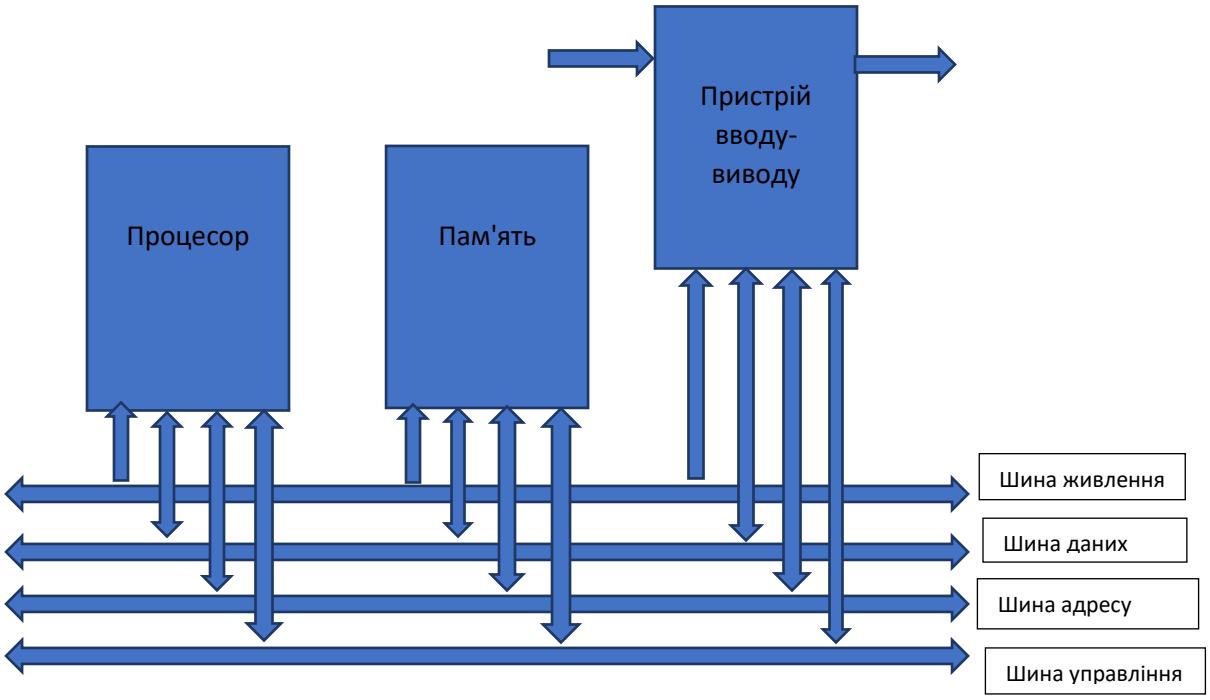



Рисунок 3.7 Структура мікропроцесорної системи

Вона включає три основні типи пристроїв:

• процесор;

• пам'ять, що включає оперативну пам'ять (RAM) та постійну пам'ять (ROM), яка служить для зберігання даних та програм;

• пристрої вводу/виводу (I/O Dev) службовці для зв'язку мікропроцесорної системи із зовнішніми пристроями.

Усі пристрої мікропроцесорної системи об'єднуються загальною системною шиною (вона називається ще системною магістраллю чи каналом). Системна магістраль включає чотири основні шини нижнього рівня:

• шина адреси (Address Bus);

• шина даних (Data Bus);
• шина управління (Control Bus);
• шина живлення (Power Bus).

Шина адреси служить визначення адреси (номера) пристрою, з яким процесор обмінюється інформацією на даний момент. Кожному пристрою (крім процесора), кожному осередку пам'яті в мікропроцесорній системі надається власна адреса. Коли код якоїсь адреси виставляється процесором на шині адреси, пристрій, якому ця адреса приписана, розуміє, що він має обмін інформацією. Шина адреси може бути односпрямованою та двоспрямованою.

Шина даних – це основна шина, яка використовується передачі інформаційних кодів між усіма пристроями мікропроцесорної системи. Зазвичай у пересиланні інформації бере участь процесор, який передає код даних в якийсь пристрій або в комірку пам'яті або приймає код даних з якогось пристрою або комірки пам'яті. Але можлива також передача інформації між пристроями без участі процесора. Шина даних завжди двоспрямована.

Шина управління на відміну шини адреси і шини даних складається з окремих керуючих сигналів. Кожен із цих сигналів під час обміну інформацією має свою функцію. Деякі сигнали служать для стробування даних, що передаються або приймаються (тобто визначають моменти часу, коли інформаційний код виставлений на шину даних). Інші сигнали, що управляють, можуть використовуватися для підтвердження прийому даних, для скидання всіх пристроїв у вихідний стан, для тактування всіх пристроїв і т.д. Лінії шини управління можуть бути односпрямованими та двоспрямованими.

Нарешті, шина живлення призначена задля пересилання інформаційних сигналів, а живлення системи. Вона складається з ліній живлення та загального дроту.

Якщо в мікропроцесорну систему треба ввести вхідний код (або вхідний сигнал), процесор по шині адреси звертається до потрібного пристрою введення/так і приймає по шині даних вхідну інформацію. Якщо з мікропроцесорної системи треба вивести вихідний код (або вихідний сигнал), процесор звертається по шині адреси до потрібного вводу/виводу і передає йому по шині даних вихідну інформацію.

Якщо інформація має пройти складну багатоступінчасту обробку, процесор може зберігати проміжні результати в системній оперативній пам'яті. Для звернення до будь-якої комірки пам'яті процесор виставляє її адресу на шину адреси і передає в неї інформаційний код по шині даних або приймає з неї інформаційний код по шині даних. У пам'яті (оперативної і постійної) знаходяться також і коди, що управляють (команди виконуваної процесором

програми), які процесор також читає по шині даних з адресацією по шині адреси. Постійна пам'ять використовується переважно для зберігання програми початкового пуску мікропроцесорної системи, яка виконується щоразу після включення живлення. Інформація до неї заноситься виробником раз і назавжди.

Таким чином, у мікропроцесорній системі всі інформаційні коди та коди команд передаються по шинах послідовно, по черзі. Це визначає порівняно невисоку швидкодію мікропроцесорної системи. Воно обмежене зазвичай навіть швидкодія процесора (яке теж дуже важливо) і швидкістю обміну по системної шині (магістралі), саме послідовним характером передачі з системної шині (магістралі).

Важливо враховувати, що пристрої введення/так найчастіше є пристроями на жорсткій логікі. Там може бути покладено частина функцій, виконуваних мікропроцесорної системою. Тому у розробника завжди є можливість перерозподіляти функції системи між апаратною та програмною реалізацією оптимальним чином. Апаратна реалізація прискорює виконання функції, але має недостатню гнучкість. Програмна реалізація значно повільніше не забезпечує високої гнучкості. Апаратна реалізація функції збільшує вартість системи та її енергоспоживання, програмна – не збільшує. Найчастіше застосовується комбінування апаратних та програмних функцій.

Іноді пристрої вводу/виводу мають у своєму складі процесор, тобто є невеликою спеціалізованою мікропроцесорною системою. Це дозволяє перекласти частину програмних функцій на пристрої введення/виведення, розвантаживши центральний процесор системи.

# 4. АРИФМЕТИЧНІ І ЛОГІЧНІ ОСНОВИ МПС

***Арифметичні основи цифрової техніки***

У цифрових пристроях доводиться мати справу з різними видами інформації. Це в чистому вигляді двійкова інформація, така як увімкнений прилад або вимкнений, справний пристрій чи ні. Інформація може бути подана у вигляді текстів, і тоді доводиться букви алфавіту кодувати за допомогою бінарних рівнів сигналу. Досить часто інформація може являти собою числа. Числа можуть бути представлені в різних системах числення. Форма записи у яких чисел істотно різниться між собою, тому, як перейти до особливостям представлення чисел в цифровий техніці, розглянемо їх запис різних системах числення.

*Системи зчислення*

Система зчислення - це сукупність прийомів та правил для представлення чисел за допомогою цифрових знаків.

Існує безліч способів запису чисел цифровими знаками, але будь-яка система числення повинна забезпечувати:

• діапазон уявлення будь-якого числа;

• єдиність уявлення (кожній комбінації символів відповідає лише одна величина).

Усі системи числення діляться на позиційні та непозиційні. У непозиційної системі числення значимість цифри у місці числа однакова, тобто. не залежить від позиції розташування. Наприклад, унарна система з одним символом, рівним одиниці. Така система числення призначена для сумарного рахунку (вузлики на «пам'ять», зарубки, рисочки, рахунок на пальцях та ін.). Для зображення якогось числа в цій системі потрібно записати число одиниць (паличок), що дорівнює цьому числу. Ця система неефективна, оскільки запис числа виходить занадто довгим.

Основа (базис) системи числення — це кількість знаків або символів, що використовуються для зображення цифр у даній системі числення.

Можливо безліч позиційних систем числення, так як за основу можна прийняти будь-яке число і утворити нову систему числення.

Інший приклад "майже непозиційної" системи чисельності - римська система рахунка. У римській системі рахунки використовуються такі символи:

I – 1; V – 5; X – 10; L - 50; C – 100; D - 500; M - 1000.

Правила переведення з римської системи чисельності до арабської системи полягають у наступному. Менша за величиною цифра, що стоїть

праворуч від більшої цифри, складається з більшої, а менша за величиною цифра, що стоїть наліво від більшої цифри, віднімається від більшої.

Позиційні системи чисельності — це системи чисельності, у яких значення цифри запису числа N залежить від її позиції (місця). Наприклад, у десятковій системі чисельність число 05 означає п'ять одиниць, 50 означає п'ять десятків, 500 – п'ять сотень і т.д.

Основа (базис) системи чисельності - це кількість знаків або символів, що використовуються для зображення цифр у даній системі чисельності.

Можливо безліч позиційних систем чисельності, тому що за основу можна прийняти будь-яке число і створити нову систему чисельності.

Переклад у позиційних системах числення

Переведення в десяткову систему числення. Будь-яке число N у позиційній системі числення можна подати у вигляді полінома

Для переведення в десяткову систему обчислюємо таку суму.

Наприклад, число 253,2410 у звичайній десятковій формі можна представити так:

$$Л^{10} = 253,24\ (10) = 2 \cdot 10^2 + 5 \cdot 10^1 + 3 \cdot 10^{\circ} + 2 \cdot 10^{-1} + 4 \cdot 10^{-2}.$$

Приклад. Двійкове число 1101,01(2) перевести до десяткової системи числення.

У двійковій системі числення для представлення чисел використовуються дві цифри 0 І 1 І двійкове ЧІСЛО $1101,01_2$ можна визначити так:

$$ТУ_2 = 1101,01_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^{\circ} + 0 \cdot 2^{-|} + 1 \cdot 2^{-2} =$$

$$=8 + 4 + 0+1+0+1/4= 14,25_{10}.$$

Якщо за правилами десяткової арифметики виконати дії у правій частині наведеної рівності, то отримаємо десятковий еквівалент двійкового числа:

$$1101,01_2 = 8 + 4 + 0+ 1 +0 + 1/4 = 14,25_{10}.$$

Переведення чисел з десяткової системи числення довільну систему числення з основою цілої частини десяткового числа полягає в наступному. Цілу частину десяткового числа необхідно послідовно ділити на ц (підстава довільної системи числення) до того часу, поки десяткове число стане рівним нулю. Залишки, отримані при розподілі та записані в послідовності, починаючи з останнього залишку, є цифрами числа ^-річної системи числення.

Правила переведення дробової частини десяткового числа є наступними. Дробну частину десяткового числа необхідно послідовно множити на (підставу довільної системи) і відокремлювати цілу частину до того часу, поки вона стане рівної нулю чи буде досягнуто задана точність перекладу.

Цілі частини результатів множення у порядку, що відповідає їх отриманню, становлять число у новій системі.

## *Логічні основи цифрової техніки*

### *Основні відомості з алгебри логіки*

Математичний апарат, на основі якого здійснюється опис цифрових схем, - це алгебра логіки, або, булева алгебра. У практичних цілях першим застосував її американський вчений Клод Шеннон під час дослідження електричних кіл із контактними вимикачами.

Логіка - наука, що вивчає методи доказів та спростування, тобто. методи встановлення істинності чи хибності одних висловлювань (тверджень, суджень) з урахуванням істинності чи хибності інших висловлювань.

Алгебра логіки також називається обчисленням висловлювань.

Висловлювання — це якась пропозиція, щодо якої можна стверджувати, що вона чи істинна, чи хибна, та інших можливостей немає («принцип двозначності — виключеного третього»).

Висловлювання відрізняються від інших мовних утворень тим, що їм можна надати лише два значення: значення «істина», якщо вони істинні; значення "брехня", якщо вони помилкові.

Якщо логіка має справу із змістом висловлювань, то алгебри логіки працюють із формулами. Будь-який елементарний вислів позначається буквою латинського алфавіту.

Приклади простих висловлювань

A=<2 плюс 2 буде 4> - це справжнє висловлювання;

?=<Кишинів - столиця Румунії> - це хибне висловлювання.

З простих висловлювань можна будувати складніші, застосовуючи звані зв'язки.

Оскільки висловлювання може набувати одне з двох значень, то говорять про «змінні висловлювання». Заміщення змінною конкретним висловлюванням означає надання одного із значень – «істина» чи «брехня».

З елементарних висловлювань з допомогою логічних зв'язок «і», «чи», «не», «якщо: те» та інших. (логічних операцій) будуються складні висловлювання — формули (чи функції) логіки алгебри.

Логічні зв'язки – це функція алгебри логіки (ФАЛ), аргументами якої є прості висловлювання.

Способи побудови нових висловлювань із заданих за допомогою логічних зв'язок, їх перетворення та встановлення істинності вивчаються у логіці висловлювань за допомогою методів алгебри.

Базовими елементами, якими оперує алгебра логіки, є висловлювання.

Висловлювання будуються над безліччю, що складається всього з двох елементів: B = {Брехня, Істина}. Над елементами даної множини визначено три операції:

• заперечення (унарна операція);
• кон'юнкція (бінарна), диз'юнкція (бінарна);
• логічний нуль та логічна одиниця – константи.

Алгебра логіки розглядає логічні вирази як алгебраїчні, які можна перетворити за певними правилами. У алгебри виразах логіки змінні є логічними (0 і 1). Знаки операцій означають логічні операції (логічні зв'язки).

Зв'язки, що існують між логічними операціями, відображають закони булевої алгебри. Сформулюємо основні їх.

Аксіоми.

A1. а) $x + y = y + x$; б) $x * y = y * x$;

A2. а) $(x + y) + z = x + (y + z)$; б) $(x * y) * z = x * (y * z)$;

A3. а) $(x + y) * z = x * z + y * z$; б) $(x * y) + z = (x + z) * (y + z)$;

A4. а) $x + x = x$; б) $x * x = x$.

Правило старшинства логічних операцій встановлює черговість виконання логічних операцій на логічних функціях, тобто. логічні операції мають пріоритет і виконуються в наступній черговості:

• одиночне заперечення;
• кон'юнкція;
• диз'юнкція, додавання за модулем 2 тощо;
• загальне заперечення.

*Функції алгебри логіки*

• Логічною функцією називається функція $P(X_1,X_2, X_3, ..., Xp ..., Xп)$, яка, так само як і її аргументи, може набувати лише двох значень (0 і 1).

• Розглянемо деякий набір аргументів $X_1$, $X_2$, $X_3$,..., Xp..., Xп} і вважатимемо, що кожен із аргументів набуває лише одного з двох можливих значень: 0 або 1.

• Так як функція на кожному наборі може прийняти значення 0 або 1, а число різних наборів 2", то загальна кількість різних функцій аргументів є $2^2$ .

• Логіка висловлювань стала основним математичним інструментом під час створення комп'ютерів. Вона легко перетворюється на бітову логіку: істинність висловлювання позначається одним бітом (0 - БРЕХНЯ, 1 - ІСТИНА).

• Згодом алгебра булева була відокремлена від логіки висловлювань шляхом введення характерних для логіки висловлювань аксіом. Це дозволило

розглядати, наприклад, логіку кубитів, потрійну логіку (коли є три варіанти істинності висловлювання: «істина», «брехня» та «не визначено»), комплексну логіку та ін.

• Математичний апарат логіки алгебри використовується для конструювання електронних пристроїв, так як основною системою числення в комп'ютері є двійкова система.

• Особливістю алгебри логіки є застосування для опису роботи дискретних пристроїв обчислювальної техніки. Одні й самі пристрої комп'ютера можуть застосовуватися обробки і зберігання як числової інформації, поданої в двійковій системі числення, і логічних змінних.

• Дані та команди подаються у вигляді двійкових послідовностей різної структури та довжини.

***Логічні елементи***

• Логічні елементи — пристрої, призначені для виконання логічних функцій (операцій) над вхідними сигналами (операндами, даними), що надаються у цифровій формі.

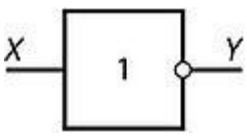
• Логічні елементи виконують логічні операції «І», «АБО», «НЕ» та комбінації цих операцій. Зазначені логічні операції можна реалізувати за допомогою контактно-релейних та електронних схем. В даний час в переважній більшості застосовують електронні логічні елементи, причому електронні логічні елементи входять до складу мікросхем. Маючи у розпорядженні логічні елементи «І», «АБО», «НЕ», можна сформулювати цифровий електронний пристрій будь-якої складності. Електронна частина комп'ютера складається з логічних елементів.

• Розглянемо логічні елементи з одним входом – елемент «НЕ» та «Повторювач» (буфер).

• Логічний елемент НЕ (інвертор)

• Елемент, що виконує функцію інверсії «НЕ», має один вхід та один вихід. Він змінює рівень сигналу протилежний. Низький потенціал на вході дає високий потенціал на виході і навпаки.

• Умовно-графічне позначення інвертора на важливих схемах у вітчизняній документації.




*Логічний елемент "Повторення"*

Елемент, що виконує функцію «Повторення», має один вхід та один вихід. Він змінює рівень сигналу на протилежний. Низький потенціал на вході дає

низький потенціал на виході, високий на вході потенціал дає високий потенціал на виході.

На важливих схемах логічний елемент «Повторення» позначають так:


*Логічний елемент "І"*

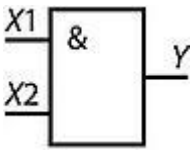
На малюнку представлена таблиця істинності елемента І з двома входами. Добре видно, що логічна одиниця з'являється на виході елемента лише за наявності одиниці першому та другому входах. У трьох інших випадках на виході будуть нулі.

| Вхід *X* | Вхід *X2* | Вихід *У* |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

Логічний елемент, що реалізує функцію кон'юнкції, називається схемою збігу. Мнемонічне правило для кон'юнкції з будь-якою кількістю входів звучить так: на виході буде:

• «1» тоді і лише тоді, коли на всіх входах діють «1»;

• «О» тоді і лише тоді, коли хоча б на одному вході діє «О». На важливих схемах логічний елемент «І» позначають так:




*Логічний елемент «АБО»*

Елемент «АБО» з двома входами працює дещо по-іншому. Досить логічної одиниці першому чи другому вході, як у виході буде логічна одиниця. Дві одиниці також дадуть одиницю на виході.

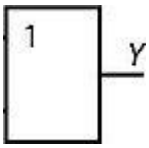
| Вхід *XI* | Вхід *X2* | Вихід *У* |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

Мнемонічне правило для диз'юнкції з будь-якою кількістю входів звучить так: на виході буде:

• «1» тоді і лише тоді, коли хоча б на одному вході діє «1»;

• «О» тоді і лише тоді, коли на всіх входах діють «О».

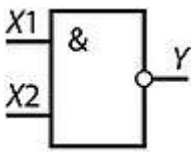
На схемах елемент «АБО» зображують так:




*Логічний елемент "2І-НЕ" (штрих Шеффера)*

У логічному елементі "2І-НЕ" цифра завжди позначає кількість входів логічного елемента. У разі це двовходовой елемент «І», вихідний сигнал якого інвертується, тобто. «О» перетворюється на «1», а «1» перетворюється на «О». Звернімо увагу на кружальце на виходах - це символ інверсії.

На принципових схемах логічний елемент «2І-НЕ» позначають так:




*Логічний елемент «Виключає АБО-НЕ» (рівнозначність)*

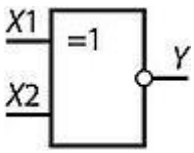
До базових логічних елементів прийнято відносити елемент, що реалізує функцію «Виключає АБО-НЕ». Інакше ця функція називається "Рівнозначність".

Якщо на виході логічного елемента "Виключає АБО-НЕ" є інвертор, то функція виконується протилежна - "Рівнозначність". Високий потенціал на виході виникає лише у тому випадку, якщо вхідні сигнали рівні.

Таблиця істинності:

| Вхід *X* | Вхід *XI* | Вихід *У* |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

Ці логічні елементи знаходять своє застосування у суматорах. «Виключає АБО-НЕ» зображується на схемах таким чином:




За принципом дії всі логічні пристрої поділяються на два класи:

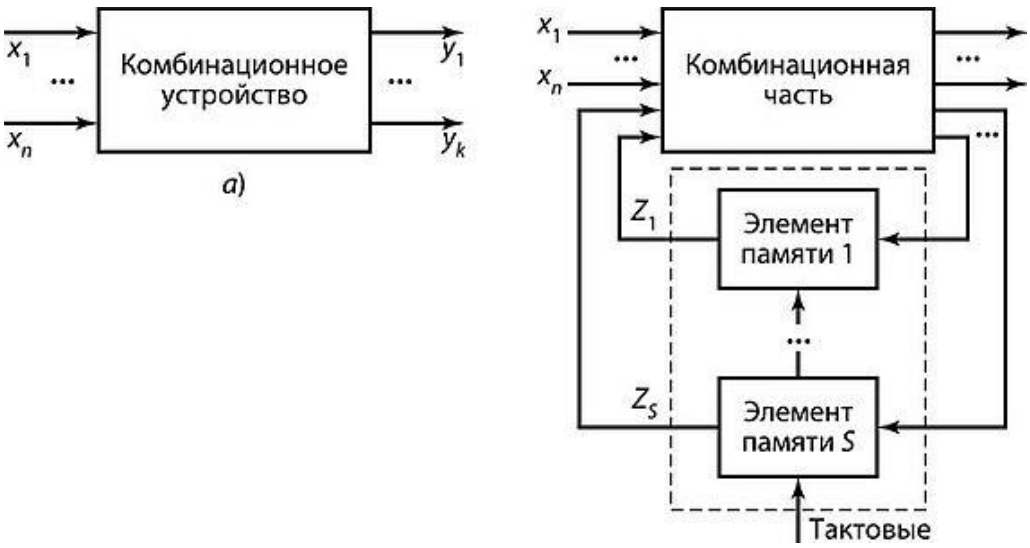
• комбінаційні;

• послідовні.29/09/22

Комбінаційними пристроями або автоматами без пам'яті називають логічні пристрої, вихідні сигнали яких однозначно визначаються тільки комбінацією змінних, що діє зараз на вході, і не залежать від значень змінних, що діяли на вході раніше.

Послідовними пристроями або автоматами з пам'яттю називають логічні пристрої, вихідні сигнали яких визначаються не тільки діє на даний момент на вході комбінацією змінних, але і всією послідовністю вхідних змінних, що діяли в попередні моменти часу. Цей тип пристроїв часто називають цифровими автоматами.

Структура послідовного та комбінаційного пристрою наведена на рис. 4.1.




*а)*

*б)*

Рисунок 4.1 Структура цифрового пристрою: а - комбінаційного; б - послідовного

У комбінаційних пристроях взаємозв'язок між вхідними та вихідними змінними задається таблицею істинності, а форма алгебри цих зв'язків описується системою рівнянь.

У послідовних пристроях вихідні змінні у,-залежать не тільки від вхідних сигналів ХТ, але і від сигналів елементів пам'яті, що надходять за цей же такт. Цифрові пристрої з пам'яттю називають кінцевими автоматами, оскільки пристрій може зберігати інформацію тільки протягом обмеженого числа тактів. Для опису роботи послідовних пристроїв використовуються таблиці переходів станів.

Відповідно до принципів побудови довільної таблиці істинності отримаємо схему суматора за модулем 2. Ця схема наведена на рис. 4.2.

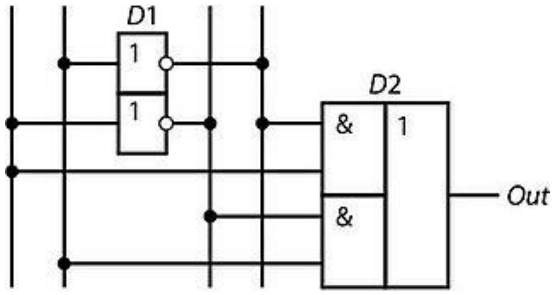
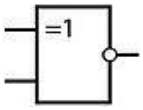



Рисунок 4.2 Принципова схема, що реалізує таблицю істинності суматора за модулем 2

Суматор по модулю 2 (схема виключає «АБО») зображується на схемах наступним чином:




Суматор модуля 2 виконує підсумовування без урахування перенесення. У звичайному двійковому суматорі потрібно враховувати перенесення, тому потрібні схеми, що дозволяють формувати перенесення до наступного двійкового розряду.

***Послідовні пристрої***

*Тригери*

Найпростіша схема, що дозволяє запам'ятовувати двійкову інформацію, будується на основі найпростіших логічних елементів "АБО" або "І". Таку схему, побудовану на елементах «І», наведено на рис. 4.3. Вхід S (Set) дозволяє встановлювати вихід тригера Q одиничний стан при подачі на його вхід логічного нуля. Вхід R (Reset) дозволяє скидати вихід тригера Q в нульовий стан при подачі з його вхід логічного нуля.

Сумматор по модулю 2 выполняет суммирование без учета переноса. В обычном двоичном сумматоре требуется учитывать перенос, поэтому требуются схемы, позволяющие формировать перенос в следующий двоичный разряд.

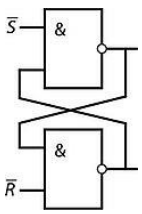



Рисунок 4.3 Схема найпростішого тригера на схемах "І". Входи R та S інверсні (активний рівень нуль).

Схема тригера дозволяє запам'ятовувати стан логічної схеми, але оскільки у початковий час може виникати перехідний процес (у цифрових схемах цей процес називається небезпечні гонки), то запам'ятовувати стани логічної схеми необхідно лише у певні моменти часу, коли всі перехідні процеси закінчені, т.к. е. Цифрові схеми вимагають синхросигналу. Усі перехідні процеси мають закінчитися під час періоду синхросигналу. Для таких цифрових схем потрібні

синхронні тригери. Схема синхронного тригера наведено на рис. 4.4, а позначення на важливих схемах - на рис. 4.5.

У наведеній схемі для запису логічного нуля та логічної одиниці потрібні різні входи, що завжди зручно. Тому для запам'ятовування дискретної інформації використовуються D-тригери.

D-тригер (від англ, delay - затримка) запам'ятовує стан входу та видає його на вихід. D-тригери мають як мінімум два входи.

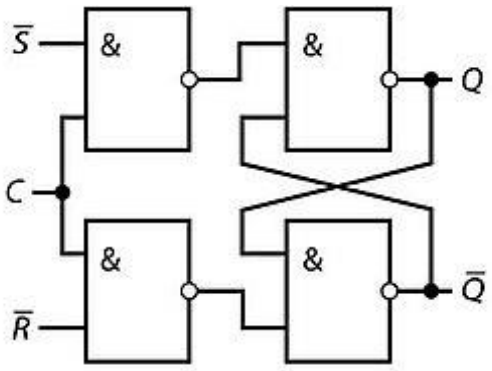



Рисунок 4.4 Схема синхронного тригера на схемах And

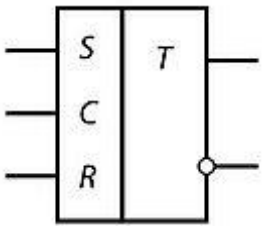



Рисунок 4.5 Умовно-графічне позначення синхронного тригера

***Реєстри***

*Реєстр зберігання.*

Регістр — внутрішній пристрій процесора або зовнішнього пристрою, призначений для тимчасового зберігання оброблюваної або керуючої інформації. Регістри є сукупністю тригерів, кількість яких дорівнює розрядності регістру, і допоміжних схем, які забезпечують виконання деяких елементарних операцій. Набір цих операцій залежно від функціонального призначення регістру може включати одночасну установку всіх розрядів регістра в нуль, паралельне або послідовне завантаження регістра, зсув вмісту регістра вліво або вправо на необхідне число розрядів, керовану видачу інформації з регістра (зазвичай використовується при роботі декількох загальну шину даних) і т.д.

Регістри зберігання використовуються для приймання, зберігання та видачі багато розрядного коду. Вони є сукупністю одноступінчастих тригерів

(як правило, /)-типу) із загальним входом синхронізації. Іноді в регістрі є також загальний вхід асинхронної установки всіх тригерів в нуль.

*Реєстр зсуву.*

Регістр зсуву - регістр, що забезпечує, крім зберігання інформації, зсув вліво або вправо всіх розрядів одночасно на однакову кількість позицій. При цьому розряди, що висуваються за межі регістру, губляться, а до розрядів, що звільняються, заноситься інформація, що надходить по окремому зовнішньому.

### *Пристрої, що запам'ятовують*

*Постійні запам'ятовуючі пристрої (ПЗП)*

Дуже часто у різних застосуваннях потрібне зберігання інформації, яка не змінюється у процесі експлуатації пристрою. Це така інформація, як програми в мікроконтролерах, початкові завантажувачі та ВПБ у комп'ютерах, таблиці коефіцієнтів цифрових фільтрів у сигнальних процесорах. Майже завжди ця інформація не потрібна одночасно, тому найпростіші пристрої для запам'ятовування постійної інформації можна побудувати на мультиплексорах.

Пам'ять сучасних універсальних комп'ютерів повинна мати інформаційну ємність $10^7...10^{12}$ байт при часі вибірки 2...20 нс. Ці параметри повинні поєднуватися з високою надійністю, низькою вартістю, малим енергоспоживанням та прийнятними масою та габаритними розмірами. На рівні розвитку техніки неможливо реалізувати пам'ять з необхідними параметрами як єдиного устрою, тому ЗУ сучасних обчислювальних систем мають багаторівневу ієрархічну структуру. Безпосередньо пов'язані із процесором модулі пам'яті становлять верхні рівні ієрархії. Вони мають максимальну швидкодію, але щодо малу інформаційну ємність. На інших рівнях ієрархії модулі пам'яті розташовуються зі збільшенням інформаційної ємності і пов'язаного з цим зменшення швидкодії. Запам'ятовувачі верхніх рівнів ієрархічної пам'яті утворюють внутрішню пам'ять, а ЗУ нижніх рівнів – зовнішню пам'ять комп'ютерів.

### *Арифметико-логічний пристрій*

Усі операції, що виконуються в арифметико-логічному пристрої, є логічними операціями (функціями), які можна розділити на наступні групи:

- операції двійкової арифметики для чисел із фіксованою точкою;
- операції двійкової (або шістнадцяткової) арифметики для чисел з плаваючою точкою;
- операції десяткової арифметики;
- операції індексної арифметики (при модифікації адрес команд);
- операції спеціальної арифметики;

• операції з логічними кодами (логічні операції);
• операції над алфавітно-цифровими полями.

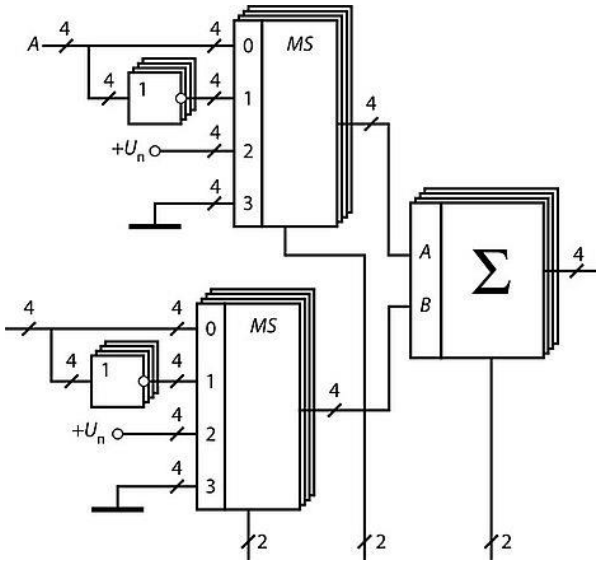



Рисунок 4.6 Структурна схема арифметичного пристрою

Сучасні комп'ютери загального призначення зазвичай реалізують операції всіх наведених вище груп, а малі та мікроЕОМ, мікропроцесори та спеціалізовані ЕОМ часто не мають апаратури арифметики чисел з плаваючою точкою, десяткової арифметики та операцій над алфавітно-цифровими полями. І тут ці операції виконуються спеціальними підпрограмами.

До арифметичних операцій відносяться додавання, віднімання, віднімання модулів («короткі операції») і множення та розподіл («довгі операції»). Групу логічних операцій складають операції диз'юнкція (логічне АБО) та кон'юнкція (логічне І) над багато розрядними двійковими словами, порівняння кодів на рівність. Спеціальні арифметичні операції включають нормалізацію, арифметичний зсув (зсуваються тільки цифрові розряди, знаковий розряд залишається на місці), логічний зсув (знаковий розряд зсувається разом з цифровими розрядами). Велика група операцій редагування алфавітно-цифрової інформації. Кожна операція в АЛУ є логічною функцією або послідовністю логічних функцій, що описуються двійковою логікою для двійкових ЕОМ, трійковою логікою для трійкових ЕОМ, чотирирічною логікою для четверичних ЕОМ, десятковою логікою для десяткових ЕОМ і так далі.

# 5 ПРОГРАМНИЙ ОБМІН ІНФОРМАЦІЄЮ В МПС

Мікропроцесорна система забезпечує більшу гнучкість роботи, вона здатна налаштовуватися на будь-яке завдання. Ця гнучкість обумовлена насамперед тим, що функції, що виконуються системою, визначаються програмою (програмним забезпеченням, software), яку виконує процесор. Апаратура (апаратне забезпечення, hardware) залишається незмінною за будь-якого завдання. Записуючи в пам'ять системи програму, можна змусити мікропроцесорну систему виконувати будь-яке завдання, яке підтримує дана апаратура. До того ж шинна організація зв'язків мікропроцесорної системи дозволяє досить легко замінювати апаратні модулі, наприклад, замінювати пам'ять на нову більшого обсягу або вищої швидкодії, додавати або модернізувати пристрої вводу/виводу, нарешті замінювати процесор більш потужний. Це дозволяє збільшити гнучкість системи, продовжити її життя за будь-якої зміни вимог до неї.

Гнучкість мікропроцесорної системи визначається не лише цим. Налаштовуватися завдання допомагає ще й вибір роботи системи, тобто режиму обміну інформацією системної магістралі (шині).

Практично будь-яка розвинена мікропроцесорна система (у тому числі й комп'ютер) підтримує три основні режими обміну магістраллю:

- програмний обмін інформацією;
- обмін із використанням переривань (Interrupts);
- обмін із використанням прямого доступу до пам'яті (DMA).

***Програмний обмін інформації*** є основним у будь-якій мікропроцесорній системі. Він передбачений завжди, без нього неможливі інші режими обміну. У цьому режимі процесор є одноосібним власником (або задатчиком, Master) системної магістралі. Усі операції (цикли) обміну інформацією у разі ініціюються лише процесором, вони виконуються суворо у порядку, запропонованому виконуваної програмою.

Процесор читає (вибирає) з пам'яті коди команд і виконує їх, читаючи дані з пам'яті або пристрою вводу/виводу, обробляючи їх, записуючи їх, записуючи дані в пам'ять або передаючи їх у пристрій введення/виводу. Шлях процесора за програмою може бути лінійним, циклічним, може містити переходи (стрибки), але завжди безперервний і повністю перебуває під контролем процесора. На жодні зовнішні події, не пов'язані з програмою, процесор не реагує (рис. 5.1). Усі сигнали магістралі у разі контролюються процесором.

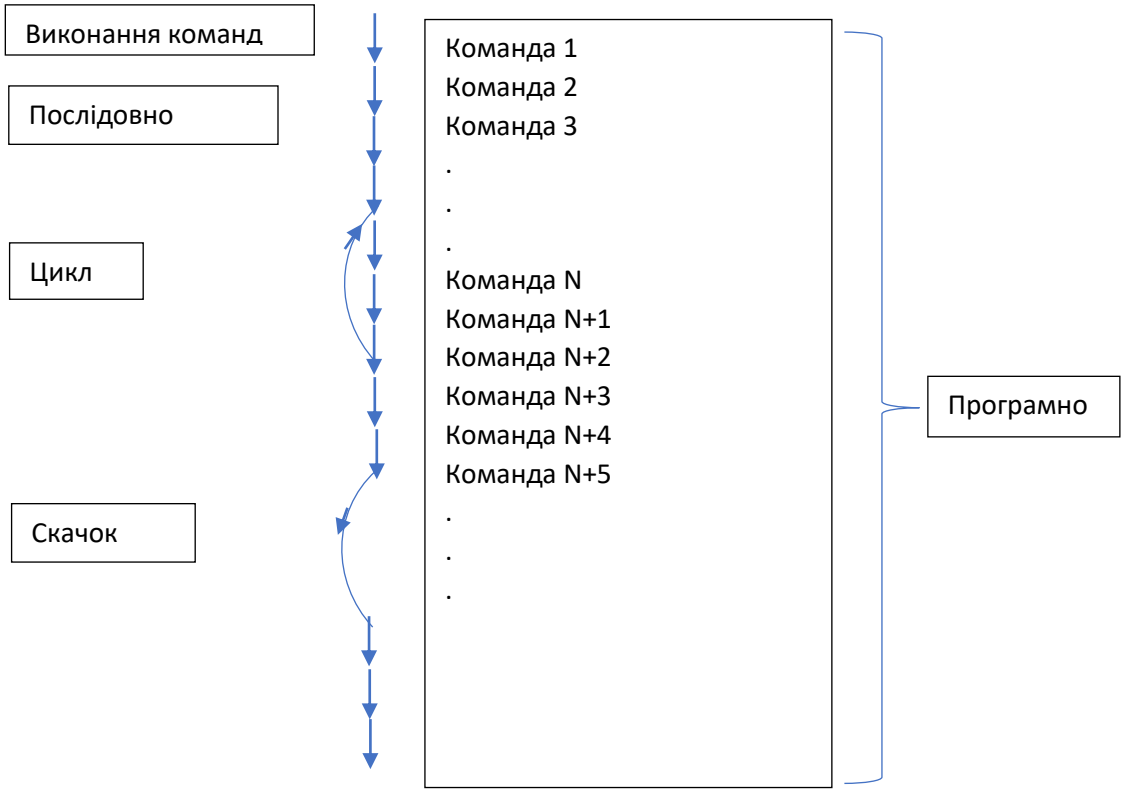



Рисунок 5.1 Програмний обмін інформацією

Обмін перериванням використовується тоді, коли необхідна реакція мікропроцесорної системи на якусь зовнішню подію, на прихід зовнішнього сигналу. У випадку комп'ютера зовнішнім подією може бути, наприклад, натискання клавіші клавіатури або прихід локальної мережі пакета даних. Комп'ютер повинен реагувати на це, відповідно, виведенням символу на екран або читанням та обробкою прийнятого по мережі пакета.

У випадку організувати реакцію на зовнішню подію можна трьома шляхами:

• За допомогою постійного програмного контролю за фактом настання події (так званого методу опитування прапора або polling);

• З допомогою переривання, тобто. насильницького переведення процесів з виконання поточної програми на виконання екстрено необхідної програми;

• За допомогою прямого доступу до пам'яті, тобто. без участі процесора за його відключення від системної магістралі.

Перший випадок з опитуванням прапора реалізується з мікропроцесорної системи постійним читанням інформації процесором з пристрою вводу/виводу, пов'язаного з зовнішнім пристроєм, на поведінку якого необхідно терміново реагувати.

У другому випадку режим переривання процесор, отримавши запит переривання від зовнішнього пристрою, закінчує виконання поточної команди і переходить до програми обробки переривання. Закінчивши виконання програми обробки переривання, він повертається до перерваної програми з точки, де його перервали (рис. 5.2).

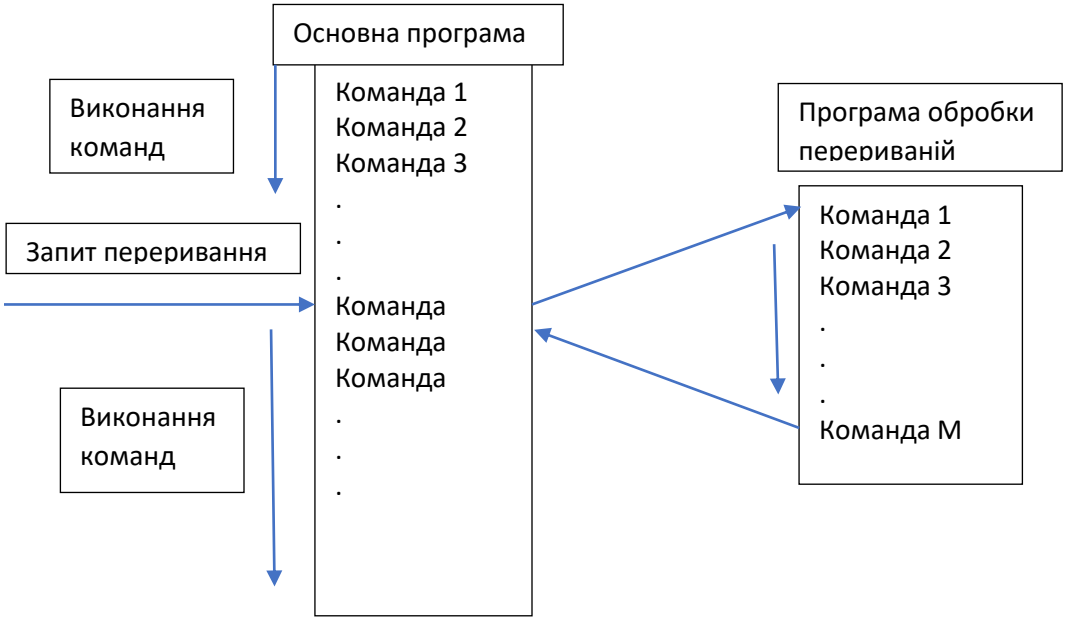



Рисунок 5.2 Обслуговування переривання

Тут важливо, що вся робота, як і у разі програмного режиму, здійснюється самим процесором, зовнішня подія просто тимчасово відволікає його. Реакція на зовнішню подію з переривання у випадку повільніше, ніж за програмному режимі. Як і у випадку програмного обміну, тут усі сигнали на магістралі виставляються процесором, тобто повністю контролює магістраль. Для обслуговування переривань в систему іноді вводиться спеціальний модуль контролера переривань, але він в обміні інформацією не бере участі. Його завдання полягає в тому, щоб спростити роботу процесора із зовнішніми запитами переривань. Цей контролер зазвичай програмно керується процесором системної магістралі.

Жодного прискорення роботи системи переривання не дає. Його застосування дозволяє лише відмовитися від постійного опитування прапора зовнішньої події та тимчасово, до настання зовнішньої події, зайняти процесор виконанням якихось інших завдань.

Прямий доступ до пам'яті – це режим, що принципово відрізняється від двох раніше розглянутих режимів тим, що обмін системною шиною йде без участі процесора. Зовнішні пристрої, що вимагають обслуговування, сигналізують процесору, що режим ПДП необхідний, у відповідь на це

процесор закінчує виконання поточної команди і відключається від усіх шин, сигналізуючи пристрій, що запитав, що обмін в режимі ПДП можна починати.

Операція ПДП зводиться до пересилання інформації з пристрою введення/виведення в пам'ять або з пам'яті пристрій введення/так. Коли пересилання інформації буде завершено, процесор знову повертається до перерваної програми, продовжуючи її з тієї точки, де його перервали. Це схоже на режим обслуговування переривань, але в даному випадку процесор не бере участі в обміні. Як і у разі переривань, реакція на зовнішню подію при ПДП значно повільніша, ніж за програмного режиму.

***Системи команд процесора***

Для створення ефективних програм необхідно мати найзагальніше уявлення про систему команд використовуваного процесора. Найкомпактніші та найшвидші програми та підпрограми створюються мовою асемблера, використання якого без знання системи команд абсолютно неможливо, адже мова асемблера є символьним записом цифрових кодів машинної мови, кодів команд процесора. Звичайно, для розробки програмного забезпечення існують різноманітні програмні засоби. Користуватися ними можна і без знання системи команд процесора. Найчастіше використовуються мови програмування високого рівня. Однак знання системи команд та мови асемблера дозволяє в кілька разів підвищити ефективність деяких найважливіших частин програмного забезпечення будь-якої мікропроцесорної системи – від мікроконтролера до персонального комп'ютера. Саме тому розглянемо основні типи команд, що є у більшості процесорів, та особливості їх застосування.

Кожна команда, що вибирається (читається) з пам'яті процесором, визначає алгоритм поведінки процесора на найближчі кілька тактів. Код команди говорить про те, яку операцію потрібно виконати процесору і з якими операндами (тобто кодами даних), де взяти вихідну інформацію для виконання команди та куди помістити результат (якщо необхідно). Код команди може займати від одного до декількох байт, причому процесор дізнається про те, скільки байт команди йому треба читати, з першого прочитаного ним байта або слова. У процесорі код команди розшифровується та перетворюється на набір мікрооперацій, що виконуються окремими вузлами процесора. Але розробнику мікропроцесорних систем це знання не надто важливе, йому важливий лише результат виконання тієї чи іншої команди.

У загальному випадку система команд процесора включає наступні чотири основні групи команд:

- команди пересилання даних;
- арифметичні команди;
- логічні команди;
- команди переходів.

Команди пересилання даних не вимагають виконання жодних операцій над операндами. Операнди просто пересилаються (точніше, копіюються) із джерела (Source) до приймача (Destination). Джерелом та приймачем можуть бути внутрішні регістри процесора, комірки пам'яті або пристрої введення/виводу. АЛУ у разі не використовується.

Арифметичні команди виконують операції складання, віднімання, множення, поділу, збільшення одиницю (інкрементування), зменшення одиницю (декрементування) тощо. Цим командам потрібно один або два вхідні операнди. Формують команди один вихідний операнд.

Логічні команди проводять над операндами логічні операції, наприклад логічне «І», логічне «АБО», що виключає «АБО», очищення, інверсію, різноманітні зрушення (вправо, вліво, арифметичний зсув, циклічний зсув). Цим командам, як і арифметичним, потрібно один чи два вхідних операнда, і вони формують один вихідний операнд.

Нарешті, команди переходів призначені зміни звичайного порядку послідовного виконання команд. З їхньою допомогою організуються переходи на підпрограми та повернення з них, всілякі цикли, розгалуження програм, пропуски фрагментів програм тощо. Команди переходів завжди змінюють вміст лічильника команд. Переходи можуть бути умовними та безумовними. Саме ці команди дозволяють будувати складні алгоритми обробки інформації.

Відповідно до результату кожної виконаної команди встановлюються або очищаються біти регістру стану процесора {PSW). Але треба пам'ятати, що не всі команди змінюють всі прапори, що є в PSW. Це визначається особливостями кожного конкретного процесора.

Розглянемо тепер особливості чотирьох виділених груп команд процесора докладніше.

*Команди пересилання даних*

Команди пересилання даних займають дуже важливе місце у системі команд будь-якого процесора. Вони виконують такі найважливіші функції:

• завантаження (запис) вмісту у внутрішні регістри процесора;
• збереження у пам'яті вмісту внутрішніх регістрів процесора;
• копіювання вмісту з однієї області пам'яті до іншої;
• запис у пристрої вводу/виводу та читання з пристроїв введення/виведення.

До команд пересилання даних також належать команди обміну інформацією (їх позначення будується на основі слова Exchange). Може бути передбачений обмін інформацією між внутрішніми регістрами, між двома половинами одного регістру (SWAP) або між регістром та осередком пам'яті.

*Арифметичні команди*

Арифметичні команди розглядають коди операндів як числові двійкові чи двійково-десяткові коди. Ці команди можуть бути поділені на п'ять основних груп:

• команди операцій з фіксованою комою (додавання, віднімання, множення, поділ);
• команди операцій з плаваючою комою (додавання, віднімання, множення, поділ);
• команди очищення;
• команди інкременту та декременту;
• команда порівняння.

Команди операцій із фіксованою комою працюють із кодами в регістрах процесора чи пам'яті як із звичайними двійковими кодами. Команда додавання (ADD) обчислює суму двох кодів. Команда віднімання (SUB) обчислює різницю двох кодів. Команда множення (MUL) обчислює добуток двох кодів (розрядність результату вдвічі більша за розрядність співмножників). Команда поділу (DIV) обчислює частки від поділу одного коду на інший. Причому всі ці команди можуть працювати з числами зі знаком, і з числами без знака.

Команди операцій з плаваючою комою (точкою) використовують формат подання чисел з порядком і мантисою (зазвичай ці числа займають два послідовні осередки пам'яті). У сучасних потужних процесорах набір команд з плаваючою комою не обмежується лише чотирма арифметичними діями, а містить і безліч інших, складніших команд, наприклад, обчислення тригонометричних функцій, логарифмічних функцій, а також складних функцій, необхідних при обробці звуку та зображення.

Команди очищення (CLR) призначені для запису нульового коду в регістр або комірку пам'яті. Ці команди можуть бути замінені командами пересилання нульового коду, але спеціальні команди очищення зазвичай виконуються швидше, ніж команди пересилання. Команди очищення іноді відносять до групи логічних команд, але їх суть від цього змінюється.

Команди інкременту (збільшення на одиницю, INC) та декременту (зменшення на одиницю, DEC) також бувають дуже зручними. Їх можна в принципі замінити командами підсумовування з одиницею або віднімання одиниці, але інкремент і декремент виконуються швидше, ніж підсумовування та віднімання. Ці команди вимагають одного вхідного операнда, який одночасно є вихідним операндом.

Нарешті, команда порівняння призначена для порівняння двох вхідних операндів. По суті, вона обчислює різницю цих двох операндів, але вихідного операнда не формує, а лише змінює біти в регістрі стану процесора {PSW) за результатом цього віднімання. Наступна за командою порівняння команда (зазвичай це команда переходу) аналізуватиме біти в регістрі стану процесора і виконуватиме дії в залежності від їх значень. У деяких процесорах передбачені

команди ланцюжкового порівняння двох послідовностей операндів, що знаходяться в пам'яті.

*Логічні команди*

Логічні команди виконують над операндами логічні (побітові) операції, тобто, вони розглядають коди операндів не як єдине число, бо як набір окремих бітів. Цим вони відрізняються від арифметичних команд. Логічні команди виконують такі основні операції:

• логічне «І», логічне «АБО», додавання за модулем 2 (що виключає «АБО»);
• логічні, арифметичні та циклічні зрушення;
• перевірка бітів та операндів;
• встановлення та очищення бітів (прапорів) регістру стану процесора {PSW).

Команди логічних операцій дозволяють бітно обчислювати основні логічні функції двох вхідних операндів. Крім того, операція «І» {AND) використовується для примусового очищення заданих бітів (як один з операндів при цьому використовується код маски, в якому розряди, що вимагають очищення, встановлені в нуль). Операція "АБО" {OR) застосовується для примусової установки заданих бітів (як один з операндів при цьому використовується код маски, в якому розряди, що вимагають установки в одиницю, рівні одиниці). Операція «що виключає АБО» {XOR) використовується для інверсії заданих бітів (як один з операндів при цьому застосовується код маски, в якому біти, що підлягають інверсії, встановлені в одиницю). Команди вимагають двох вхідних операндів та формують один вихідний операнд.

Команди зрушень дозволяють побітно зрушувати код операнду вправо (у бік молодших розрядів) чи ліворуч (убік старших розрядів). Тип зсуву (логічний, арифметичний або циклічний) визначає, яке буде нове значення старшого біта (при зрушенні вправо) або молодшого біта (при зрушенні вліво), а також визначає, чи буде десь збережено колишнє значення старшого біта (при зрушенні вліво) або молодшого біта (при зрушенні праворуч). Наприклад, при логічному зрушенні праворуч у старшому розряді коду операнду встановлюється нуль, а молодший розряд записується як прапор переносу в регістр стану процесора. А при арифметичному зрушенні праворуч значення старшого розряду зберігається колишнім (нулем або одиницею), молодший розряд також записується як прапор переносу.

Циклічні зрушення дозволяють зрушувати біти коду операнду по колу (за годинниковою стрілкою при зрушенні праворуч або проти годинникової стрілки при зрушенні вліво). При цьому кільце зсуву може входити або не входити прапор переносу. У біт прапора перенесення (якщо він використовується) записується значення старшого біта при циклічному зсуві вліво та молодшого біта при циклічному зрушенні вправо. Відповідно, значення біта прапора

перенесення переписуватиметься в молодший розряд при циклічному зрушенні вліво і старший розряд при циклічному зрушенні вправо.

*Команди переходів*

Команди переходів призначені в організацію різноманітних циклів, розгалужень, викликів підпрограм і т.д., тобто. вони порушують послідовний перебіг виконання програми. Ці команди записують в регістр - лічильник команд нове значення і тим самим викликають перехід процесора не до наступної команди, а до будь-якої іншої команди в пам'яті програм. Деякі команди переходів передбачають повернення назад, у точку, з якої було зроблено перехід, інші не передбачають цього. Якщо повернення передбачено, поточні параметри процесора зберігаються в стеку. Якщо повернення не передбачене, поточні параметри процесора не зберігаються.

Команди переходів без повернення поділяються на дві групи:
• команди безумовних переходів;
• команди умовних переходів.

У позначках цих команд використовуються слова Branch (розгалуження) та Jump (стрибок).

Команди безумовних переходів викликають перехід на нову адресу незалежно від чого. Вони можуть викликати перехід на вказану величину зсуву (вперед або назад) або на вказану адресу пам'яті. Величина зміщення або нове значення адреси вказується як вхідний операнд.

Команди умовних переходів викликають перехід який завжди, лише за виконанні заданих умов. Як такі умови зазвичай виступають значення прапорів у регістрі стану процесора {PSW). Тобто умовою переходу є результат попередньої операції, яка змінює значення прапорів. Усього таких умов переходу може бути від 4 до 16. Декілька прикладів команд умовних переходів:
• перехід, якщо дорівнює нулю;
• перехід, якщо не дорівнює нулю;
• перехід, якщо є переповнення;
• перехід, якщо немає переповнення;
• перехід, якщо більший за нуль;
• перехід, якщо менше або дорівнює нулю.

Якщо умова переходу виконується, проводиться завантаження в регістр — лічильник команд нового значення. Якщо ж умова переходу не виконується, лічильник команд просто нарощується і процесор вибирає та виконує наступну по порядку команду.

Спеціально для перевірки умов переходу застосовується команда порівняння (CMP), що передує команді умовного переходу (або навіть кільком командам умовних переходів). Але прапори можуть встановлюватися і будь-якою іншою командою, наприклад, командою пересилання даних, будь-якою

арифметичною або логічною командою. Зазначимо, що самі команди переходів прапори не змінюють, що й дозволяє ставити кілька команд переходів одну за одною.

Спільне використання кількох команд умовних та безумовних переходів дозволяє процесору виконувати розгалужені алгоритми будь-якої складності. На рис. 5.3 показано розгалуження програми на дві гілки з наступним з'єднанням, а на рис. 5.4 - розгалуження на три гілки з наступним з'єднанням.

Команди переходів з подальшим поверненням у точку, з якої було зроблено перехід, застосовуються до виконання підпрограм, тобто. допоміжних програм. Ці команди також називаються командами виклику підпрограм (поширена назва — CALL). Використання підпрограм дозволяє спростити структуру основної програми, зробити її більш логічною, гнучкою, легкою для написання та налагодження. У той самий час слід враховувати, що широке використання підпрограм, зазвичай, збільшує час виконання програми.

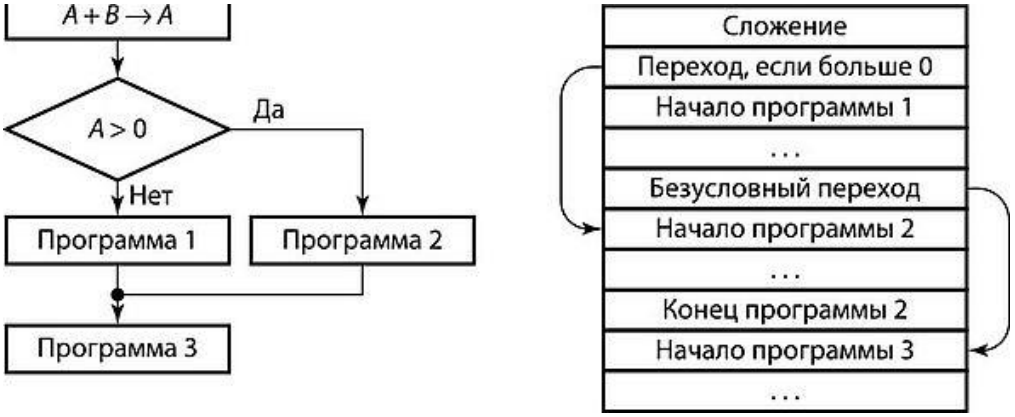



Рисунок 5.3 Реалізація розгалуження програми на 2 гілки

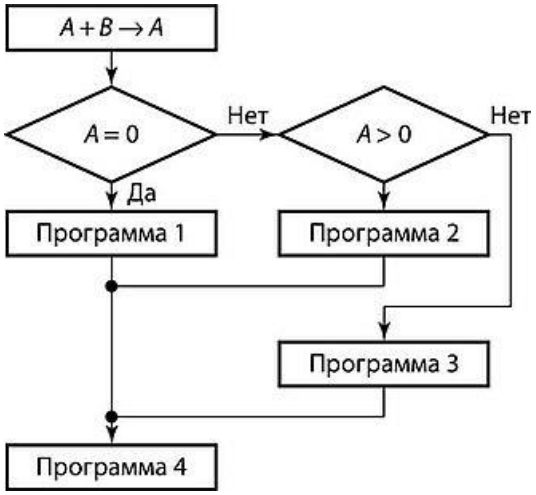



Рисунок 5.4 Реалізація розгалуження програми на 3 гілки

Усі команди переходів із поверненням передбачають безумовний перехід (вони не перевіряють жодних прапорів). При цьому вони вимагають одного вхідного операнду, який може вказувати абсолютне значення нової адреси, так і зсув, що складається з поточним значенням адреси. Поточне значення лічильника команд (поточна адреса) зберігається перед виконанням переходу до стеку.

Для повернення до точки виклику підпрограми (точку переходу) використовується спеціальна команда повернення (RET або RTS). Ця команда отримує з стека значення адреси команди переходу і записує його в регістр - лічильник команд.

Особливе місце серед команд переходу із поверненням займають команди переривань (поширена назва – INT). Ці команди в якості вхідного операнду вимагають номер переривання (адреса вектору). Обслуговування таких переходів здійснюється так само, як і апаратних переривань. Тобто для виконання цього переходу процесор звертається до таблиці векторів переривань і отримує з неї за номером переривання адресу пам'яті, в яку йому необхідно перейти. Адреса виклику переривання та вміст регістру стану процесора (PSW) зберігаються у стеку. Збереження SW – важлива відмінність команд переривання від команд переходів із поверненням.

Команди переривань у багатьох випадках виявляються зручнішими, ніж звичайні команди переходів із поверненням. Сформувати таблицю векторів переривань можна один раз, а потім уже звертатися до неї за необхідності. Номер переривання відповідає номер підпрограми, тобто. номеру функції, що виконується підпрограмою. Тому команди переривання набагато частіше включаються до систем команд процесорів, ніж звичайні команди переходів із поверненням.

Для повернення з підпрограми, викликаної командою переривання, використовується команда повернення з переривання (IRET або RTI). Ця команда витягує зі стека збережене значення лічильника команд і регістру стану процесора {PSW).

Зазначимо, що в деяких процесорів передбачені команди умовних переривань, наприклад команда переривання при переповненні.

*Адресація операндів*

Більшість команд процесора працює з кодами даних (операндами). Одні команди вимагають вхідних операндів (одного чи двох), інші видають вихідні операнди (найчастіше один операнд). Вхідні операнди називаються ще операндами-джерелами, а вихідні називаються операндами-приймачами. Всі ці коди операндів (вхідні та вихідні) повинні десь розташовуватися. Вони можуть знаходитися у внутрішніх регістрах процесора (найбільш зручний та швидкий варіант). Вони можуть розташовуватися в системній пам'яті (найпоширеніший

варіант). Нарешті, вони можуть перебувати в пристроях вводу/виводу (найрідкіший випадок). Визначення місця розташування операндів виконується кодом команди. Існують різні методи, за допомогою яких код команди може визначити, звідки брати вхідний операнд і куди поміщати вихідний операнд. Ці методи називаються методами адресації. Ефективність вибраних методів адресації багато чому визначає ефективність роботи всього процесора загалом.

*Методи адресації*

Кількість методів адресації у різних процесорах може бути від 4 до 16. Розглянемо кілька типових методів адресації операндів, що використовуються зараз у більшості мікропроцесорів.

Безпосередня адресація передбачає, що операнд (вхідний) перебуває у пам'яті безпосередньо за кодом команди. Операнд зазвичай є константою, яку треба кудись переслати, до чогось додати і т.д. Наприклад, команда може полягати в тому, щоб додати число 6 до якогось внутрішнього регістру процесора. Це число 6 розташовуватиметься у пам'яті, всередині програми на адресу, наступного за кодом цієї команди складання.

Пряма (або абсолютна) адресація передбачає, що операнд (вхідний або вихідний) знаходиться в пам'яті за адресою, код якої знаходиться всередині програми відразу ж за кодом команди. Наприклад, команда може полягати в тому, щоб очистити (зробити нульовим) вміст осередку пам'яті з адресою 1000000. Код цієї адреси 1000000 буде розміщуватися в пам'яті, всередині програми в наступній адресі за кодом цієї команди очищення.

Реєстрова адресація передбачає, що операнд (вхідний чи вихідний) перебуває у внутрішньому регістрі процесора. Наприклад, команда може полягати в тому, щоб переслати число з нульового регістру до першого. Номери обох регістрів (0 та 1) будуть визначатися кодом команди пересилання.

Непрямо-реєстрова (або непряма) адресація припускає, що у внутрішньому регістрі процесора знаходиться не сам операнд, а його адреса в пам'яті. Наприклад, команда може полягати в тому, щоб очистити комірку пам'яті з адресою, що знаходиться в нульовому регістрі. Номер цього регістру (0) визначатиметься кодом команди очищення.

Автоінкрементна адресація дуже близька до непрямої адресації, але відрізняється від неї тим, що після виконання команди вміст регістру, що використовується, збільшується на одиницю або на два. Цей метод адресації дуже зручний, наприклад, при послідовній обробці кодів масиву даних, що знаходиться в пам'яті. Після обробки якогось коду адреса в регістрі вказуватиме вже наступний код з масиву. При використанні непрямої адресації в цьому випадку довелося б збільшувати вміст цього регістру окремою командою.

Автодекрементна адресація працює на кшталт автоінкрементної, але тільки вміст вибраного регістру зменшується на одиницю або два перед виконанням команди. Ця адресація також зручна під час обробки масивів даних. Спільне використання автоінкрементної та автодекрементної адресацій дозволяє організувати пам'ять стекового типу.

З інших поширених методів адресації можна згадати про індексні методи, які передбачають обчислення адреси операнда додавання до вмісту регістру заданої константи (індексу). Код цієї константи розміщується у пам'яті безпосередньо за кодом команди.

Зазначимо, що вибір того чи іншого методу адресації значною мірою визначає час виконання команди. Найшвидша адресація - це реєстрова, тому що вона не вимагає додаткових циклів обміну магістраллю. Якщо ж адресація вимагає звернення до пам'яті, час виконання команди буде збільшуватися за рахунок тривалості необхідних циклів звернення до пам'яті. Зрозуміло, що чим більше внутрішніх регістрів у процесора, тим частіше і вільніше можна застосовувати реєстрову адресацію і швидше працюватиме система в цілому.

# 6 АРХІТЕКТУРА КОМП’ЮТЕРА

Обчислювальна система є результатом інтеграції апаратних засобів та програмного забезпечення, що функціонують в єдиній системі та призначених для спільного виконання інформаційно-обчислювальних процесів.

Архітектура (architecture) - це базова організація системи, втілена в її компонентах, їх відносинах між собою та з оточенням, а також принципи, що визначають проектування та розвиток системи.

Архітектура обчислювальної машини (Computer architecture) - це концептуальна структура обчислювальної машини, що визначає проведення обробки інформації та включає методи перетворення інформації в дані та принципи взаємодії технічних засобів та програмного забезпечення.

Таким чином, архітектуру обчислювальної машини можна представити як множина взаємопов'язаних компонентів, що включають елементи різної природи: програмне забезпечення (software), апаратне забезпечення (hardware), алгоритмічне забезпечення (brainware), спеціальне фірмове забезпечення (firmware), які створюють можливість обробки інформації та отримання результату у необхідній формі.

Під архітектурою обчислювальної машини розуміють загальний опис принципів організації апаратно-програмних засобів та основних їх характеристик, що визначають функціональні можливості обчислювальної машини.

Архітектура обчислювальної системи - сукупність характеристик і параметрів, що визначають функціонально-логічну та структурно-організовану систему і зачіпають в основному рівень обчислювачів, що паралельно працюють.

Поняття архітектури охоплює загальні поняття організації системи, які включають такі високорівневі аспекти розробки комп'ютера, як пам'яті, структура системної шини, організація вводу/вывода тощо.

Архітектура визначає принципи дії, інформаційні зв'язки та взаємне з'єднання основних логічних вузлів комп'ютера: процесора, оперативного запам'ятовуючого пристрою (ЗП), зовнішніх ЗП та периферійних пристроїв. Спільність архітектури різних комп'ютерів забезпечує їхню сумісність з погляду користувача.

Відповідно до всесвітньо відомого фахівця в галузі інформаційних технологій Ендрю Таненбауму в основі структурної організації комп'ютера лежить ідея ієрархічної структури, в якій кожен рівень виконує певну функцію. Це твердження однаково стосується як апаратної організації, і організації програмного забезпечення.

Переваги такого представлення обчислювальних машин:

• кожен верхній рівень інтерпретується одним чи декількома нижніми рівнями;

• кожен із рівнів можна проектувати незалежно;

• що нижчий рівень, у якому реалізується програма, тим більше висока продуктивність досяжна;

• модифікація нижніх рівнів не впливає реалізації верхніх.

Сучасні комп'ютери можна як структуру, що з шести рівнів (рис. 6.1).

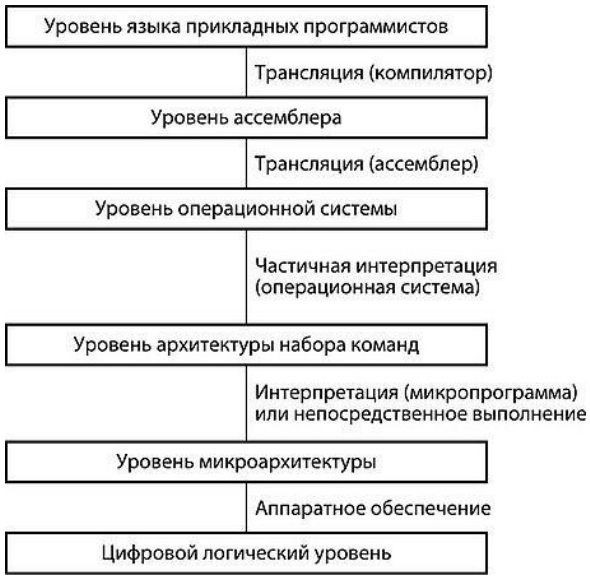



Рисунок 6.1 Структура шестирівневого комп'ютера

Спосіб підтримки кожного рівня вказаний під ним, у дужках надано назву відповідного програмного забезпечення.

Існують рівні, які розташовані нижче за нульовий рівень. Ці рівні не розглядаються через складність матеріалу, оскільки вони потрапляють у сферу таких напрямів, як елементна база мікро- та наноелектроніки, мікросистемна техніка.

Рівень 0: Цифровий логічний рівень. Цифровий логічний рівень є апаратне забезпечення комп'ютера. Об'єктами цифрового рівня є цифрові логічні пристрої. Для опису того, як функціонують цифрові логічні пристрої, використовується математичний апарат логіки алгебри, в якій використовуються логічний нуль і логічна одиниця.

Основу проектування складних цифрових пристроїв становлять функції. Базові логічні елементи - це схеми, що містять електронні ключі (вентилі) та виконують основні логічні операції. Складні логічні функції можна виразити через сукупність кінцевого числа базисних логічних функцій, як-от «НЕ», «І», «АБО».

Одним із найважливіших елементів цифрової техніки є тригер (англ. Trigger — клямка, спусковий гачок). Тригер має два стійкі стани, один з яких відповідає двійковій одиниці, а інший - двійковому нулю. Сам тригер не є базовим елементом, тому що він збирається з простіших логічних схем.

Тригер - це електронна схема, що широко застосовується в регістрах комп'ютера для надійного запам'ятовування двійкової одиниці (біта пам'яті).

Біти пам'яті, об'єднані групи, наприклад по 16, 32 або 64, формують регістри. Кожен регістр може містити одне двійкове число до певної межі.

Рівень 1: рівень мікроархітектури. У комп'ютерній інженерії мікроархітектура (англ, microarchitecture; іноді скорочується до march або uarch), також звана організацією комп'ютера, - це спосіб, яким ця архітектура набору команд (ISA, АН К) реалізована в процесорі. Кожна АН К може бути реалізована за допомогою різних мікроархітектур. Реалізації можуть змінюватись в залежності від цілей конкретної розробки або в результаті технологічних зрушень. Архітектура комп'ютера є комбінацією мікроархітектури, мікрокоду та АН До.

На цьому рівні в обробці команд бере участь арифметико-логічний пристрій (АЛП). АЛУ складається з регістрів, суматора з відповідними логічними схемами та елемента управління виконуваним процесом. Оперативна пам'ять організована як послідовностей, які групуються в машинні слова. Арифметико-логічний пристрій працює відповідно до іменами (кодами) операцій, що повідомляються йому, які при пересиланні даних потрібно виконати над змінними, що поміщаються в регістри. Над якими кодами проводиться операція, куди міститься її результат - визначається командою, що виконується. Прикладами обробки можуть бути логічні операції («І», «АБО», «Виключає АБО» і т.д.), тобто. побиті операції над операндами, і навіть арифметичні операції (складання, віднімання, множення, розподіл тощо.). Команди утримують від одного до трьох операндів.

В операційному пристрої (АЛУ) реалізується задана послідовність мікрокоманд (команд), мікропрограмному пристрої управління (УУ) задається послідовність мікрокоманд (команд).

Розрізняють два види мікрокоманд: зовнішні - такі мікрокоманди, які надходять в АЛУ від зовнішніх джерел і викликають у ньому перетворення інформації, і внутрішні - ті, які генеруються в АЛУ і впливають на мікропрограмний пристрій, змінюючи таким чином нормальний порядок проходження команд.

Мікропрограма - це інтерпретатор для команд на рівні 2. Мікропрограма викликає команди з пам'яті і виконує їх одну за одною, а результати обчислень з АЛП передаються в оперативну пам'ять кодових шин запису.

На деяких машинах робота тракту даних контролюється спеціальною програмою, яка називається мікропрограмою. На інших машинах тракт даних контролюється апаратними засобами.

Наприклад, при виконанні команди ADD мікропрограма викликає з пам'яті операнди команди додавання, поміщає їх у регістри, АЛУ обчислює суму операндів, а потім результат переправляється назад у пам'ять. На комп'ютері з апаратним контролем тракту даних відбувається така сама процедура, але при цьому немає програми, яка інтерпретує команди рівня 2.

Машини з різною мікроархітектурою можуть мати однакову архітектуру набору команд і таким чином бути придатними для виконання тих самих програм. Нові мікроархітектури та/або схемотехнічні рішення разом із прогресом у напівпровідникової промисловості є тим, що дозволяє новим поколінням процесорів досягати більш високої продуктивності, використовуючи ту ж АН К.

Рівень 2: рівень архітектури набору команд. Архітектура набору команд (англ. instruction set architecture, ISA) - частина архітектури комп'ютера, що визначає програмовану частину ядра мікропроцесора. На цьому рівні визначаються реалізовані мікропроцесори конкретного типу:

• архітектура пам'яті;
• взаємодія із зовнішніми пристроями вводу/виводу;
• режими адресації;
• регістри;
• машинні команди;
• різні типи внутрішніх даних (наприклад, з плаваючою комою, цілочисельні типи тощо);
• обробники переривань та виняткових станів.

У сучасних мікроархітектурах використовують конвеєрний тракт. Конвеєр містить такі стадії, як вибір інструкцій, декодування інструкцій, виконання та запис результату. Деякі архітектури включають також доступ до пам'яті. Дизайн конвеєра є фундаментальним для розробки мікроархітектури.

Ми визначили, що обчислювальна система (ВС) - це деяке об'єднання апаратних засобів, засобів управління апаратурою (фізичними ресурсами), засобів управління логічними ресурсами, системи програмування та прикладне програмне забезпечення.

Основою цифрових обчислювальних систем є логічні цифрові схеми, засновані на елементах, що приймають два можливі фіксовані значення - "О" і "1". Інформація в таких схемах представлена у вигляді імпульсних електричних сигналів, що мають амплітуду вище певного рівня (логічний нуль) або нижче за певний рівень (логічна одиниця). При побудові цифрової ЗС реалізовано принцип програмного управління. Суть цього принципу в наступному: цифрова схема побудована таким чином, що може вирішувати певний набір простих

завдань або виконувати певні дії (команди); комбінуючи ці дії відповідно до заданого алгоритму розв'язання складної задачі (програма), можна отримати рішення для широкого кола завдань.

*Класична архітектура обчислювальної машини*

Розмаїття сучасних обчислювальних машин дуже велике, але вони є реалізацію так званої фон-неймановской (принстонської) архітектури, представленої Джорджем фон Нейманом ще 1945 р. Розглянемо класичну архітектуру обчислювальної машини з прикладу архітектури фон Неймана.

Принцип дії обчислювальної машини полягає у виконанні програм - послідовностей арифметичних, логічних та інших операцій, що описують вирішення певного завдання.

Програма (для ЕОМ) - це впорядкована послідовність команд, що підлягає обробці.

Команда – це опис операції, яку має виконати обчислювальна машина.

Результат команди виробляється за точно визначеними для цієї команди правилами, закладеними в конструкцію комп'ютера. Електронні схеми кожного комп'ютера можуть розпізнавати та виконувати обмежений набір простих команд.

На рис. 6.2 представлена схема фон-нейманівської обчислювальної машини, на основі якої вже понад півстоліття створюються сучасні обчислювальні машини.

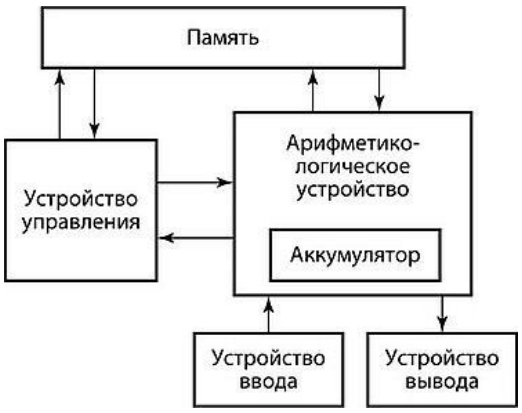



Рисунок 6.2 Схема фон-нейманівської обчислювальної машини

Фон-нейманівська архітектура складається з таких основних пристроїв:

• пам'яті, що включала 4096 машинних слів розрядністю 40 біт. Машинне слово містило або команди (дві команди по 20 біт), або ціле число зі знаком на 40 біт (8 біт вказували на тип команди, а інші 12 біт визначали одне з 212 = 4096 слів);

• арифметико-логічного пристрою (АЛП), усередині якого знаходиться особливий внутрішній регістр розрядністю 40 біт, так званий акумулятор;

• пристрої керування (УУ), виконує функції керування пристроями;

• пристрої введення інформації;
• пристрої виведення інформації.

Ці пристрої з'єднані каналами зв'язку, якими передається інформація.

У сучасних обчислювальних машинах арифметико-логічний пристрій та пристрій керування поєднуються в одній мікросхемі, яка називається центральним процесором (ЦП).

В основу побудови сучасних обчислювальних машин було покладено такі принципи фон Неймана.

1. Принцип однорідності пам'яті. Команди та дані зберігаються в одній і тій же пам'яті та зовні в пам'яті невиразні. Розпізнати їх можна лише за способом використання, тобто. одне й те значення в осередку пам'яті можна використовувати як дані, як і команда, як і адресу залежно лише від способу звернення щодо нього. Це дозволяє виконувати над командами самі операції, як і над числами, і, відповідно, відкриває ряд можливостей. Так, циклічно змінюючи адресну частину команди, можна забезпечити звернення до послідовних елементів масиву даних. Такий прийом зветься модифікації команд і з позицій сучасного програмування не вітається. Більш корисним є інше наслідок принципу однорідності, коли команди однієї програми можуть бути отримані як результат виконання іншої програми. Ця можливість є основою трансляції — перекладу тексту програми з мови високого рівня на мову конкретної обчислювальної машини.

2. Принцип адресності. Структурно основна пам'ять складається з пронумерованих осередків, причому процесору у довільний момент доступна будь-яка комірка. Двійкові коди команд і даних поділяються на одиниці інформації, звані словами, і зберігаються у осередках пам'яті, а доступу до них використовуються номери відповідних осередків — адреси.

3. Принцип програмного управління. Усі обчислення, передбачені алгоритмом розв'язання задачі, мають бути представлені у вигляді програми, що складається з послідовності керуючих слів — команд. Кожна команда наказує деяку операцію з набору операцій, що реалізуються обчислювальною машиною. Команди програми зберігаються у послідовних осередках пам'яті обчислювальної машини і виконуються у природній послідовності, тобто. у порядку їх становища у програмі. За потреби за допомогою спеціальних команд цю послідовність можна змінити. Рішення про зміну порядку виконання команд програми приймається або з аналізу результатів попередніх обчислень, або безумовно.

4. Принцип двійкового кодування. Відповідно до цього принципу, вся інформація (як дані, і команди) кодується двійковими цифрами 0 і 1. Кожен тип інформації представляється двійковою послідовністю і має свій формат. Послідовність бітів у форматі, що має певний зміст, називається полем. У

числової інформації зазвичай виділяють поле знака та поле значущих розрядів. У форматі команди можна виділити два поля: поле коду операції та поле адрес.

Величезною перевагою фон-нейманівської архітектури є її простота, тому дана концепція стала основою більшості комп'ютерів загального призначення. Однак спільне використання шини для пам'яті програм та пам'яті даних призводить до так званого вузького місця архітектури фон Неймана. Термін «вузьке місце архітектури фон Неймана» запровадив Джон Бекус у 1977 р. у своїй лекції «Чи можна звільнити програмування від стилю фон Неймана?», яку він прочитав під час вручення йому Премії Тьюринга.

Інший напрямок вдосконалення архітектури персонального комп'ютера пов'язаний із максимальним прискоренням обміну інформацією із системною пам'яттю. Саме з системної пам'яті комп'ютер читає всі команди, що виконуються, і в системній пам'яті зберігає дані. Тобто найбільше звернень процесор робить саме до пам'яті. Прискорення обміну з пам'яттю призводить до суттєвого прискорення роботи всієї системи загалом.

Але при використанні обміну з пам'яттю системної магістралі доводиться враховувати швидкісні обмеження магістралі. Системна магістраль повинна забезпечувати поєднання з великою кількістю пристроїв, тому вона повинна мати досить велику довжину; вона вимагає застосування вхідних та вихідних буферів для узгодження з лініями магістралі. Цикли обміну системною магістралі складні, і прискорювати їх не можна. Внаслідок суттєвого прискорення обміну процесора з пам'яттю по магістралі досягти неможливо.

Розробниками було запропоновано наступний підхід. Системна пам'ять підключається не до системної магістралі, а до спеціальної високошвидкісної шини, що знаходиться «ближче» до процесора, що не потребує складних буферів та великих відстаней. У такому разі обмін із пам'яттю йде з максимально можливою для даного процесора швидкістю і системна магістраль не уповільнює його. Особливо актуальним це стає через зростання швидкодії процесора.

Таким чином, структура персонального комп'ютера з одношинної, що застосовувалася лише на перших комп'ютерах, стає тришинною (рис. 6.3).

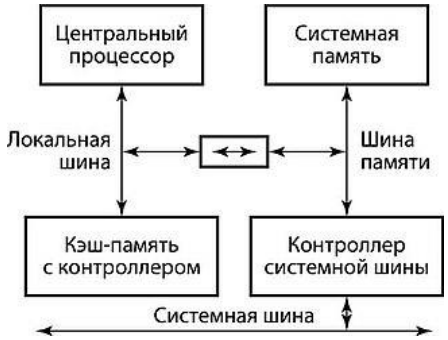



Рисунок 6.3 Організація зв'язків у тришинній структурі

Призначення шин таке:

• до локальної шини підключаються центральний процесор та кешпам'ять (швидка буферна пам'ять);

• до шини пам'яті підключається оперативна та постійна пам'ять комп'ютера, а також контролер системної шини;

• до системної шини (магістралі) підключаються всі інші пристрої комп'ютера.

Всі три шини мають адресні лінії, лінії даних та керуючі сигнали. Але склад і призначення ліній цих шин не збігаються між собою, хоча вони виконують однакові функції. З погляду процесора системна шина (магістраль) у системі всього одна, по ній він отримує дані та команди і передає дані як у пам'ять, так і пристрої вводу/виводу.

По призначенню:

• універсальні МП;

• спеціальні МП.

*Універсальні МП можуть бути використані для вирішення широкого кола завдань. При цьому їхня ефективна продуктивність мало залежить від проблемної специфіки завдання. Як правило, це визначається досить широкою універсальною системою команд.*

*Спеціальні МП - проблемно орієнтовані МП, які націлені на прискорене виконання певних функцій, що збільшує ефективну продуктивність при вирішенні лише певного завдання:*

*• математичні процесори;*

*• мікроконтролери;*

*• паралельна обробка даних;*

*• цифрова обробка сигналу – цифрові фільтри тощо.*

*3. На вигляд оброблюваних вхідних сигналів:*

*• цифрові МП;*

*• аналогові МП.*

*Вимога до аналогових МП:*

*• велика розрядність;*

*• Висока швидкість арифметичних операцій.*

4. За характером тимчасової організації роботи.

• одномагістральні - всі пристрої мають однаковий інтерфейс і підключаються до єдиної інформаційної магістралі, через яку передаються коди даних, адрес і керуючих сигналів;

• багатомагістральні – пристрої групами підключаються до своєї інформаційної магістралі, це дозволяє здійснити одночасну передачу інформаційних сигналів кількома магістралями. Продуктивність зростає.

5. За кількістю програм, що виконуються:

• однопрограмні;

• мультипрограм.

Мультипрограмні можуть одночасно виконувати кілька програм, або мають засоби для підтримки віртуальної мультипрограмності.

***Архітектури багатопроцесорних обчислювальних систем***

Персональні комп'ютери дозволяють реалізувати багато комп'ютерних технологій, починаючи від роботи в Інтернеті, і закінчуючи побудовою анімаційних тривимірних сцен. Однак існують завдання, обсяг обчислень яких перевищує можливості персонального комп'ютера. Для їх вирішення застосовуються комп'ютери з набагато більш високою швидкодією. Для отримання високої швидкодії на існуючій елементній базі використовуються архітектури, в яких процес обробки розпаралелюється і виконується одночасно на декількох обробних пристроях.

Існує три основні підходи до побудови архітектур таких комп'ютерів: багатопроцесорні, магістральні та матричні архітектури.

Архітектура простих **багатопроцесорних** систем виконується за схемою із загальною шиною. Два або більше процесорів та один або кілька модулів пам'яті розміщені на загальній шині. Кожен процесор для обміну з пам'яттю перевіряє, чи вільна шина, і, якщо вона вільна, він займає її. Якщо шина зайнята, процесор чекає, поки вона звільниться. При збільшенні кількості процесорів продуктивність системи буде обмежена пропускною спроможністю шини. Щоб вирішити цю проблему, кожен процесор забезпечує власну локальну пам'ять (Рис. 6.4), куди збожеволіють тексти виконуваних програм і локальні змінні, оброблювані даним процесором. Загальний пристрій використовується для зберігання загальних змінних і загального системного програмного забезпечення. За такої організації навантаження на загальну шипу значно знижується.


Рисунок 6.4 Архітектура багатопроцесорної обчислювальної системи із загальною шиною

Один із процесорів виділяється для управління всією системою. Він розподіляє завдання виконання програм між процесорами і керує роботою загальної шини.

Периферійний процесор здійснює обслуговування зовнішніх пристроїв під час введення та виведення інформації з загальної пам'яті. Він може бути такого ж типу, як і інші процесори, але зазвичай встановлюється спеціалізований процесор, призначений до виконання операцій управління зовнішніми пристроями.

**Магістральний** принцип є найпоширенішим при побудові високопродуктивних обчислювальних систем. Процесор такої системи має кілька функціональних обробних пристроїв, що виконують арифметичні та логічні операції, та швидку реєстрову пам'ять для зберігання даних, що обробляються. Дані, лічені з пам'яті, розміщуються в регістрах і їх завантажуються в обробні пристрої. Результати обчислень перешкодять регістри і використовуються як вихідні дані для подальших обчислень. Таким чином, виходить конвеєр перетворення даних: регістри - обробні пристрої - регістри - …. Архітектура магістрального суперкомп'ютера наведена на рис. 6.5. Число функціональних пристроїв дорівнює шести («Складання», «Множення» і т.д.), проте в реальних системах їх кількість може бути іншою. Пристрій планування послідовності виконання команд розподіляє дані, що зберігаються в регістрах, на функціональні пристрої та здійснює запис результатів знову в регістри. Кінцеві результати обчислень записуються в загальний пристрій.


Рисунок 6.5 Архітектура магістрального суперкомп'ютера

У **матричній** обчислювальній системі процесори поєднуються в матрицю процесорних цементів. Як процесорні елементи можуть використовуватися універсальні процесори, що мають власний пристрій управління, або обчислювачі, що містять тільки АЛУ і виконують команди зовнішнього пристрою управління. Кожен процесорний елемент забезпечений локальною пам'яттю, що зберігає оброблювані процесором дані, але при необхідності процесорний елемент може здійснювати обмін зі своїми сусідами або із загальним пристроєм, що запам'ятовує. У першому випадку програми та дані кількох завдань або незалежних частин одного завдання завантажуються в локальну пам'ять процесорів і виконуються паралельно. У другому варіанті всі процесорні елементи одночасно виконують ту саму команду, що надходить від пристрою обробки команд на всі процесорні елементи, але над різними даними, що зберігаються в локальній пам'яті кожного процесорного елемента. Варіант архітектури із загальним управлінням показаний на рис. 6.6.

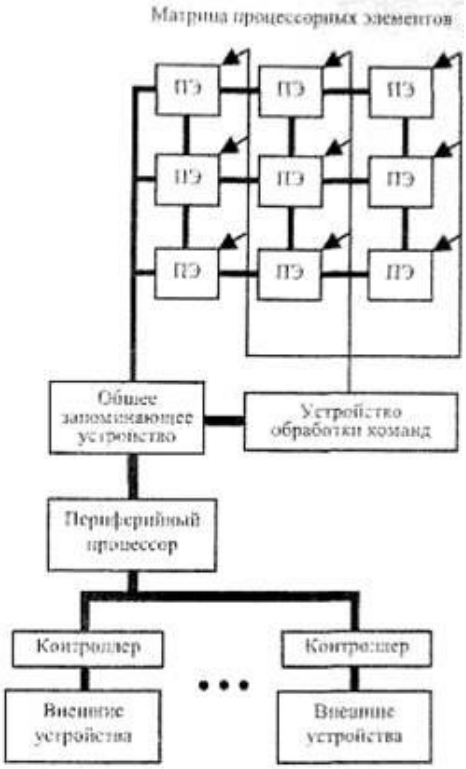



Рисунок 6.6 Архітектура матричної обчислювальної системи з загальним управлінням

Обмін даними з периферійними пристроями виконується через периферійний процесор, підключений до загального пристрою.

# 7 СИСТЕМНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРА

Комп'ютери - універсальні пристрої для обробки інформації. І тому потрібно скласти для комп'ютера зрозумілою йому мовою точну і докладну послідовність інструкцій, тобто. програму, як треба опрацьовувати інформацію. Сам по собі комп'ютер не має знання в жодній області свого застосування, всі ці знання зосереджені у програмах, що виконуються на комп'ютері. Тому часто вживане вираз «комп'ютер зробив» означає рівно те, що на комп'ютері була виконана програма, яка дозволила виконати відповідні дії.

Змінюючи програми для комп'ютера, можна перетворити його на робоче місце практично будь-якого фахівця. Під час виконання програми можуть використовувати різні пристрої для введення та виведення даних.

Таким чином, для ефективного використання комп'ютера необхідно знати призначення та властивості необхідних для роботи з ним програм.

*Системне програмне забезпечення. Поняття та функції*

Сукупність програм, призначена на вирішення завдань на ПК (персональному комп'ютері), називається програмним забезпеченням. Склад програмного забезпечення ПК називають програмною конфігурацією.

Програми, що працюють на комп'ютері, можна розділити на кілька категорій:

· Прикладні програми, що безпосередньо забезпечують виконання необхідних користувачам робіт: редагування текстів, малювання картинок, обробка інформаційних масивів.

· Інструментальні системи (системи програмування, що забезпечують створення нових програм на комп'ютері).

· Системні програми, що виконують різні допоміжні функції, наприклад: створення копій інформації, що використовується, видачу довідкової інформації про комп'ютер, перевірка працездатності пристроїв комп'ютера.

Таким чином, системне програмне забезпечення (System Software) комплекс програм, які забезпечують управління компонентами комп'ютерної системи, такими як процесор, оперативна пам'ять, пристрої введення-виведення, мережеве обладнання, виступаючи як «міжшаровий інтерфейс», з одного боку якого апаратура інший - програми користувача.

На відміну від прикладного програмного забезпечення, системне не вирішує конкретні практичні завдання, а лише забезпечує роботу інших програм, надаючи їм сервісні функції, що абстрагують деталі апаратної та мікропрограмної реалізації обчислювальної системи, керує апаратними ресурсами обчислювальної системи.

*Системне програмування – створення системного програмного забезпечення.*

Віднесення того чи іншого програмного забезпечення до системного умовно і залежить від угод, що використовуються в конкретному контексті. Як правило, до системного програмного забезпечення відносяться операційні системи, утиліти, системи програмування, системи управління базами даних, широкий клас програмного забезпечення.

*Види системного програмного забезпечення:*

Базове складається із мінімального набору програмних засобів, які займаються забезпеченням роботи комп'ютера. Базове програмне забезпечення вже постачається разом із комп'ютером. Для можливості базового програмного забезпечення обов'язково потрібне сервісне забезпечення програми і програмний комплекс для організації кращого і зручнішого місця для роботи користувача.

Сервісне (програми та програмні комплекси, які розширюють можливості базового ПЗ та організують зручніше середовище для роботи користувача).

*Утиліти*

Утиліта (англ. utility або tool) – допоміжна комп'ютерна програма у складі загального програмного забезпечення для виконання спеціалізованих типових завдань, пов'язаних із роботою обладнання та операційної системи (ОС).

Утиліти надають доступ до можливостей (параметрів, налаштувань, установок), недоступних без їх застосування, або роблять процес зміни деяких параметрів простішим (автоматизують його).

Утиліти можуть входити до складу операційних систем, йти в комплекті зі спеціалізованим обладнанням або розповсюджуватися окремо.

Залежно від операційної системи утиліти можна поділити на дві великі групи:

• Незалежні утиліти, що не потребують своєї роботи операційної системи.

• Системні утиліти, що входять у постачання ОС та потребують її наявності.

Також всі утиліти можна розділити за їх функціями, а саме:

· Диспетчери файлів (стандартна програма Windows, простий однопанельний менеджер файлів)

· Архіватори (з можливим стисненням даних);

· Переглядачі (комп'ютерна програма, призначена для перегляду файлів).

· Утиліти для діагностики апаратного чи програмного забезпечення;

· Утиліти відновлення після збоїв;

· Оптимізатор диска (вид утиліти для оптимізації розміщення файлів на дисковому накопичувачі, наприклад, шляхом дефрагментації диска);

· Деінсталятор (програма для видалення програмного забезпечення);

· Утиліти управління процесами.

Утиліти використовуються для моніторингу показників датчиків та продуктивності обладнання (наприклад, моніторингу температур процесора або відеоадаптера), керування параметрами обладнання (обмеження максимальної швидкості обертання CD-приводу; зміна швидкості обертання вентиляторів), контролю показників (перевірка цілісності; правильності запису даних); можливостей (форматування або перерозмітка диска зі збереженням даних, видалення без відновлення).

*Системи програмування*

Системою програмування називається комплекс програм, призначений автоматизації програмування завдань на ЕОМ. Система програмування звільняє проблемного користувача або прикладного програміста від необхідності написання програм вирішення своїх завдань незручною йому мовою машинних команд, і надають можливість використовувати спеціальні мови вищого рівня. Для кожної з таких мов, які називаються вхідними або вихідними, система програмування має програму, що здійснює автоматичний переклад (трансляцію) текстів програми з вхідної мови на мову машини. Зазвичай система програмування містить описи мов програмування, програми-транслятори з цих мов, а також розвинену бібліотеку стандартних підпрограм.

До цієї категорії належать системні програми, призначені для розробки програмного забезпечення:

· асемблери - комп'ютерні програми, здійснюють перетворення програми у формі вихідного тексту мовою асемблера в машинні команди у вигляді об'єктного коду;

· транслятори - програми або технічні засоби, що виконує трансляцію програми;

. компілятори - програми, що перекладають текст програми мовою високого рівня, в еквівалентну програму машинною мовою.

· Інтерпретатори - програми (іноді апаратні засоби), що аналізують команди або оператори програми і тут же виконують їх;

· компонувальники (редактори зв'язків) - програми, які проводять компонування - приймають на вхід один або кілька об'єктних модулів і збирають за ними модуль, що здійснюється;

· відладчики (англ. debugger) - модулі середовища розробки або окремі програми, призначені для пошуку помилок у програмах;

· текстові редактори - комп'ютерні програми, призначені для створення та зміни текстових файлів, а також їх перегляду на екрані, виведення на друк, пошуку фрагментів тексту тощо;

· спеціалізовані редактори вихідних текстів - текстові редактори для створення та редагування вихідного коду програм. Спеціалізований редактор вихідних текстів може бути окремим додатком, або вбудований в інтегроване середовище розробки;

· бібліотеки підпрограм - збірники підпрограм або об'єктів, що використовуються для розробки програмного забезпечення;

· редактори графічного інтерфейсу.

*Системи управління базами даних*

Система управління базами даних (СУБД) - спеціалізована програма (частіше комплекс програм), призначена для організації та ведення бази даних.

Оскільки системи управління базами даних є обов'язковим компонентом обчислювальної системи, найчастіше їх відносять до системного програмного забезпечення. Часто СУБД здійснюють лише службову функцію під час роботи інших видів програм (веб-сервери, сервери додатків), тому не завжди можна віднести до прикладного програмного забезпечення. Тому СУБД іноді належать до проміжного програмного забезпечення (Middleware)

Основні функції СУБД:

· Керування даними у зовнішній пам'яті (на дисках);

· Керування даними в оперативній пам'яті з використанням дискового кешу;

· Журналізація змін, резервне копіювання та відновлення бази даних після збоїв;

· Підтримка мов баз даних (мова визначення даних, мова маніпулювання даними).

Класифікація СУБД за способом доступу до бази даних:

· Файл-серверні, у яких файли даних розташовуються централізовано на файл-сервері, а програмна реалізація СУБД розміщується кожному клієнтському комп'ютері цілком. Доступ до даних здійснюється через локальну мережу.

· Синхронізація читань та оновлень здійснюється за допомогою файлових блокувань.

· Клієнт-серверні СУБД складаються з клієнтської частини (яка входить до складу прикладної програми) та сервера.

· Вбудовані - програмні бібліотеки, які дозволяють уніфікованим чином зберігати великі обсяги даних на локальній машині.

*Операційні системи*

Операційна система, скор. ОС (англ. operating system, OS) - комплекс керуючих і обробних програм, які, з одного боку, виступають як інтерфейс між пристроями обчислювальної системи та прикладними програмами, а з іншого боку - призначені для керування пристроями, керування обчислювальними процесами, ефективного розподілу обчислювальних ресурсів між обчислювальними процесами та організації надійних обчислень. Це визначення стосується більшості сучасних операційних систем загального призначення.

Місце ОС у структурі апаратно-програмних засобів комп'ютера показано на рис. 7.1. Нижній рівень структури становлять інтегральні мікросхеми, джерела живлення, дисководи та інші фізичні пристрої.

<table>
<tr><td colspan="3">Прикладні програми</td></tr>
<tr><td>Інтерпретатори команд</td><td>Компілятори</td><td>Редактори</td></tr>
<tr><td colspan="3">Операційна система</td></tr>
<tr><td colspan="3">Система команд</td></tr>
<tr><td colspan="3">Функціональні засоби</td></tr>
<tr><td colspan="3">Апаратні засоби</td></tr>
</table>

Рисунок 7.1 Структура апаратно-програмних засобів комп’ютера

У логічній структурі типової обчислювальної системи операційна система займає становище між пристроями з їхньою мікроархітектурою, машинною мовою і, можливо, власними (вбудованими) мікропрограмами - з одного боку - і прикладними програмами з іншого.

Розробникам програмного забезпечення операційна система дозволяє абстрагуватися від деталей реалізації та функціонування пристроїв, надаючи мінімально необхідний набір функцій

У більшості обчислювальних систем операційна система є основною, найбільш важливою (іноді й єдиною) частиною системного програмного забезпечення. Найбільш поширеними операційними системами є системи сімейства Windows та системи класу UNIX (особливо Linux та Mac OS).

Основні функції операційних систем:

· Завантаження додатків в оперативну пам'ять та їх виконання.

· Стандартизований доступ до периферійних пристроїв (пристрою введення-виведення).

· Управління оперативною пам'яттю (розподіл між процесами, організація віртуальна пам'ять).

· Управління доступом до даних на енергонезалежних носіях (таких як жорсткий диск, компакт-диск тощо), організованим у тій чи іншій файловій системі.

· Користувальницький інтерфейс.

· Мережеві операції, підтримка стека протоколів.

Є програми обчислювальної техніки, котрим операційні системи зайві. Наприклад, вбудовані мікрокомп'ютери містяться сьогодні в багатьох побутових приладах, автомобілях (іноді по десятку в кожному), стільникових телефонах і т. п. Найчастіше такий комп'ютер постійно виконує лише одну програму, яка запускається після вмикання. І прості ігрові приставки -- також спеціалізовані мікрокомп'ютери - можуть обходитися без операційної системи, запускаючи при включенні програму, записану на вставленому в пристрій

«картриджі» або компакт-диску. Тим не менш, деякі мікрокомп'ютери та ігрові приставки все ж таки працюють під управлінням особливих власних операційних систем. Найчастіше — це UNIX-подібні системи (останнє особливо вірно щодо програмованого комутаційного устаткування: файрволів, маршрутизаторів).

Системне програмне забезпечення є базою для будь-якого комп'ютера, це невід'ємна і важлива його частина. До складу системного програмного забезпечення входять такі частини, як операційна система (найпопулярніші - ОС сімейства Windows та системи класу UNIX), утиліти, системи програмування та системи управління базами даних (СУБД). Всі разом ці частини забезпечують управління компонентами комп'ютерної системи, такими як процесор, оперативна пам'ять, пристрої введення-виводу, мережеве обладнання, виступаючи як «міжшаровий інтерфейс», з одного боку апаратура, а з іншого - додатки користувача. Системне не вирішує конкретні практичні завдання, лише забезпечує роботу інших програм, надаючи їм сервісні функції, абстрагуючі деталі апаратної та мікропрограмної реалізації обчислювальної системи, управляє апаратними ресурсами обчислювальної системи.

*Базові поняття операційних систем*

Для операційних систем існує набір базових понять, таких як процеси, пам'ять та файли, які є найважливішими для розуміння загальної ідеї побудови та функціонування ОС.

Ключове поняття ОС – процес. Процесом називають програму на момент її виконання. З кожним процесом зв'язується його адресний простір – список адрес пам'яті від деякого мінімуму до деякого максимуму. За цими адресами процес може занести інформацію та прочитати її. Адресний простір містить саму програму, дані до неї та її стек. З будь-яким процесом пов'язується деякий набір регістрів, включаючи лічильник команд, покажчик стека та інші апаратні ресурси, і навіть вся інформація, необхідна запуску програми.

Щоб краще розібратися в понятті процесу, проведемо аналогію із системою, що працює в режимі поділу часу. Припустимо, ОС вирішує зупинити роботу одного процесу і запустити інший, тому що перший витратив відведену йому частину робочого часу ЦП. Пізніше зупинений процес повинен бути знову запущений з того ж стану, в якому його зупинили. Отже, всю інформацію про процес потрібно десь зберегти. Так, процес може мати кілька одночасно відкритих файлів. Пов'язаний із кожним файлом покажчик дає поточну позицію, тобто. номер байта або запису, які будуть прочитані після повторного запуску процесу. У разі тимчасового припинення дії процесу всі покажчики потрібно зберегти так, щоб команда читання, виконана після відновлення процесу, прочитала правильні дані. У багатьох ОС вся інформація про кожен процес зберігається у таблиці операційної системи. Ця таблиця називається таблицею

процесів і є пов'язаний список структур, по одній на кожен існуючий на даний момент процес.

У кожному комп'ютері є оперативна пам'ять, що використовується для зберігання програм, що виконуються. У найпростіших ОС у певний час у пам'яті може лише одна програма. Більш складні системи дозволяють одночасно зберігати у пам'яті кілька програм. Для того. Щоб вони не заважали один одному, потрібний захисний механізм. Цей механізм керується операційною системою.

Інше важливе, пов'язане з пам'яттю питання – управління адресним простором процесів. Зазвичай під кожний процес відводиться кілька адрес, які він може використовувати. У найпростішому випадку, коли максимальна величина адресного простору для процесу менша за оперативну пам'ять, процес заповнює свій адресний простір, і пам'яті вистачає на те, щоб утримувати її повністю. Однак, що станеться, якщо адресний простір процесу виявиться більшим, ніж ОЗУ комп'ютера, а процес захоче використати її? У цьому випадку використовується метод, званий віртуальною пам'яттю, при якому ОС зберігає частину адрес в оперативній пам'яті, а частина на диску і змінює їх місцями за потреби. Управління пам'яттю – важлива функція ОС.

*Файлова система* - ще одне базове поняття, яке підтримується віртуально всіма ОС. Як було встановлено, основною функцією ОС є маскування особливостей роботи дисків та інших пристроїв та надання користувачеві зрозумілою та зручною абстрактною моделлю незалежних від пристроїв файлів. Системні дзвінки потрібні для створення, видалення, читання або запису файлів. Перед тим, як прочитати файл, його потрібно розмістити на диску та відкрити, а після прочитання його потрібно закрити. Усі ці функції здійснюють системні дзвінки.

При створенні місця зберігання файлів ОС використовують поняття каталогу як способу об'єднання файлу групи. Для створення та видалення каталогу також потрібні системні виклики. Вони забезпечують переміщення існуючого файлу в каталог і видалення файлу з каталогу. Вміст каталогу може містити файли або інші каталоги. Ця модель створює структуру - файлову систему.

Кожен файл в ієрархії каталогів можна визначити, задавши його ім'я шляху, зване повним ім'ям файлу. Шлях починається з вершини структури каталогів, яка називається кореневим каталогом. Така абсолютна назва шляху складається зі списку каталогів, які потрібно пройти від кореневого каталогу до файлу, з розділенням окремих компонентів.

***Процеси та потоки***

Основним поняттям, пов'язаним з операційними системами, є абстрактне поняття, що описує роботу програми.

У багатозадачній системі процесор перемикається між програмами, надаючи кожній від десятків до сотень мілісекунд. При цьому в кожен момент часу процесор зайнятий лише однією програмою, але за секунду він встигає попрацювати з кількома програмами, створюючи у користувачів паралельну ілюзію з усіма програмами. Іноді в цьому випадку говорять про *псевдопаралелізм,* на відміну від справжнього паралелізму в *багатопроцесорних* системах, що містять кілька процесорів, що розділяють спільну пам'ять між собою.

Виробники операційних систем розробили концептуальну модель послідовних процесів, що спрощує спостереження за роботою паралельних процесів.

У моделі процесу все що функціонують комп'ютері ПЗ організовано як набору *послідовних процесів*, чи навіть процесів. Процесом є програма, що виконується разом з поточними значеннями лічильника команд, регістрів і змінних. З позицій цієї абстрактної моделі кожен процес має власний центральний віртуальний процесор. Насправді центральний процесор перемикається з процесу на процес, але для кращого розуміння системи простіше розглядати набір процесів, що йдуть паралельно, ніж представляти процесор, що перемикаються від програми до програми. Це перемикання і називається *багатозадачність* або *мультипрограмування*.

Операційної системи потрібен спосіб створення та переривання процесів у міру необхідності. Зазвичай під час завантаження ОС створюються кілька процесів. Деякі з них забезпечують взаємодію з користувачем та виконують задану роботу. Інші процеси є фоновими. Вони пов'язані з конкретними користувачами, але виконують особливі функції.

Процеси можуть створюватися у момент завантаження системи. Так, поточний процес може створити один чи кілька нових процесів, у своїй поточний процес виконує системний запит створення нового процесу. Створення нових процесів особливо корисно в тих випадках, коли завдання, що виконується, найпростіше сформувати як набір пов'язаних, але незалежно взаємодіючих процесів. Якщо необхідно організувати вибірку великої кількості даних із мережі для подальшої обробки, зручно створити один процес для вибірки даних та розміщення їх у буфері, інший – для зчитування та обробки даних із буфера. Така схема навіть прискорить обробку даних, якщо кожен процес запустити на окремому процесорі у разі мікропроцесорної системи.

Як правило, процеси завершуються у міру виконання своєї роботи. Так після закінчення компіляції програми компілятор виконує системний запит, щоб повідомити ОС про закінчення роботи. У текстових редакторах, браузерах та інших програм такого типу є кнопка або пункт меню, за допомогою яких можна завершити процес.

Процес є незалежним об'єктом зі своїм лічильником команд та внутрішнім станом, проте існує необхідність взаємодії з іншими процесами. Наприклад, вихідні дані одного процесу можуть бути вхідними даними для іншого процесу.

Модель процесів спрощує уявлення про внутрішню поведінку системи. Деякі процеси запускають програми, які виконують команди, введені користувачем з клавіатури. Інші процеси є частиною системи та обробляють такі завдання, як виконання запитів файлової служби, керування запуском диска або магнітного накопичувача.

Розглянутий підхід описується моделлю, наведеною на рис. 7.2. Нижній рівень ОС – це планувальник – невелика програма. На верхніх рівнях розташовані процеси. Обробка переривань та процедури, пов'язані із зупинкою та запуском процесів, виконуються планувальником. Решта ОС структурована як набору процесів.

| Процеси | | | | |
|---|---|---|---|---|
| 0 | 1 | … | n-2 | n-1 |
| Плановальник | | | | |

Рисунок 7.2 Нижній рівень ОС, який відповідає за переривання і планування

Реалізація моделі процесів базується на таблиці з одним елементом для кожного процесу. Елемент таблиці містить інформацію про стан процесу, лічильник команд, розподіл пам'яті, стан відкритих файлів, про покажчик стека, використання та розподіл ресурсів, а також всю іншу інформацію, яку необхідно зберігати при переключенні в стан готовності або блокування для подальшого запуску процесу, як якщо би він не зупинявся.

*Потоки*

У звичайних ОС процес визначається відповідним адресним простором та одиночним керуючим потоком. Але часто зустрічаються ситуації, коли в одному адресному просторі бажано мати кілька квазіпаралельних процесів, що управляють.

Модель процесу базується на двох незалежних концепціях: *групування ресурсів* та *виконання програми*. Коли їх поділяють, виникає поняття *потоку*.

З одного боку, процес можна як спосіб об'єднання родинних ресурсів в одну групу. Процес має адресний простір, що містить програму, дані та інші ресурси. Ресурсами є відкриті файли, дочірні процеси, аварійні невідпрацьовані повідомлення, обробники сигналів, облікова інформація та багато іншого. Набагато простіше керувати ресурсами, поєднавши їх у формі процесу.

З іншого боку, процес можна як потік виконуваних команд. Потік має лічильник команд, який відстежує порядок виконання дій. Він має регістри, у

яких зберігаються поточні змінні. Він має стек, що містить протокол виконання процесу, де на кожну викликану процедуру відведено певну структуру. Хоча потік протікає всередині процесу, слід розрізняти концепції потоку та процесу. Процеси використовуються для угруповання ресурсів, а потоки є об'єктами, що почергово виконуються на ЦП.

Концепція потоків додає до моделі процесу можливість одночасного виконання у тому самому середовищі процесу кілька досить незалежних програм.

Необхідність потоків продемонструємо на прикладі редагування дипломної роботи. Уявімо, що користувач видалив пропозицію на першій сторінці, а потім виправив пропозицію на 150 сторінці, в якій 200 сторінок. Він дає команду програмі перейти на сторінку з номером 150. Текстовому процесору доведеться переформатувати весь документ до 150 сторінки, оскільки він не знає, де починається ця сторінка. Це може зайняти чимало часу.

У цьому випадку допоможуть потоки. Нехай текстовий процесор написаний як двопоточної програми. Один потік взаємодіє з користувачем, а другий переформатує документ у фоновому режимі. Як тільки пропозиція на першій сторінці була видалена, інтерактивний потік дає команду потоку фону переформатувати весь документ. Коли перший потік продовжує виконувати команди з клавіатури або миші, другий потік швидко переформатує документ. Може статися, що форматування буде закінчено раніше, ніж користувач захоче перейти до 150 сторінці, і тоді команда буде виконана миттєво.

*Управління пам'яттю*

Пам'ять є важливим ресурсом, що вимагає ретельного управління, оскільки програми збільшуються в розмірах швидше, ніж пам'ять.

Пам'ять у комп'ютері має ієрархічну структуру. Невелика її частина є дуже швидкою енергозалежною (що втрачає інформацію при вимиканні живлення) кеш-пам'ять. Комп'ютери мають також десятки гігабайт енергозалежної оперативної пам'яті та десятки терабайт повільної енергонезалежного простору на жорсткому диску. Одним із завдань ОС є координація використання всіх цих складових пам'яті.

Частина операційної системи, що відповідає за керування пам'яттю, називається модулем керування пам'яттю або менеджером пам'яті. Менеджер стежить за тим, яка частина пам'яті використовується в даний момент, виділяє пам'ять процесам і після їхнього завершення звільняє ресурси, управляє обміном даних між ОЗП та диском.

Системи керування пам'яттю ділять на два класи. До першого класу відносяться системи, що переміщають між оперативною пам'яттю і диском під час його виконання, тобто. здійснює підкачування процесів цілком (swapping) або посторінково (paging). Звичайний та посторінковий варіанти підкачування є штучними процесами, викликаними відсутністю достатньої кількості

оперативної пам'яті для одночасного зберігання всіх програм. До другого – ті, що цього не роблять. Другий клас систем простіший. Оскільки програмне забезпечення зростає швидше, ніж пам'ять, потреба в ефективному управлінні існуватиме завжди.

Найпростіша схема управління пам'яттю - *однозадачна система без підкачування на диск* - полягає в тому, що в кожний момент часу працює тільки одна програма, і пам'ять розділяється між програмами та операційною системою. Коли система організована таким чином, у кожний конкретний момент може працювати лише один процес.

Більшість сучасних систем дозволяє одночасно запуск декількох процесів. Наявність декількох процесів, що працюють в той самий момент часу, означає, що, коли один процес припинено в очікуванні операції введення-виведення, інший може використовувати центральний процесор. Багатозадачність збільшує завантаження процесора.

Інша стратегія, що має назву віртуальної пам'яті, що дозволяє програмам працювати навіть тоді, коли вони лише частково знаходяться в оперативній пам'яті. Основна ідея віртуальної пам'яті полягає в тому, що об'єднаний розмір програми, даних та стека може перевищувати кількість доступної фізичної пам'яті. Операційна система зберігає програми, що використовуються зараз в оперативній пам'яті, інші – на диску. При цьому частини програми, що знаходяться на диску та в пам'яті, будуть змінюватися місцями за необхідності.

*Ввід-вивід*

Однією з найважливіших функцій ОС є керування пристроями введення-виведення комп'ютера. Пристрої введення-виводу поділяють на дві категорії: блокові та символьні пристрої. Блокові пристрої (наприклад диски) зберігають інформацію у вигляді блоків фіксованого розміру, причому кожен блок має свою адресу. Символьні пристрої (принтери, мережні адаптери, миші) приймає або репрезентує потік неструктурованих символів. Операційна система дає цим пристроям команди, перехоплює переривання та обробляє помилки. Вона повинна забезпечити простий і зручний інтерфейс між пристроями та іншою частиною системи. Інтерфейс повинен бути однаковим для всіх пристроїв з метою досягнення незалежності від апаратури, що застосовується. Програмне забезпечення вводу-виводу становить істотну частину операційної системи.

# 8 СИСТЕМИ ВВОДУ-ВИВОДУ ІНФОРМАЦІЇ

Комп'ютер є унікальним інструментом для суттєвого підвищення швидкості обробки інформації, інтелектуальних можливостей технічних систем, підвищення якості систем управління. Найважливішою функцією комп'ютера є можливість взаємодії із зовнішнім світом, що включає збирання інформації щодо об'єкта дослідження та видачі сигналів, що впливають на керований об'єкт.

Важливість системи вводу/виводу визначається ще й тим, що швидке збільшення продуктивності процесорів настільки змінило принципи класифікації комп'ютерів, що саме з організації вводу/виводу ми можемо грубо їх відрізняти. Різниця між мейнфреймом і міні комп’ютером у тому, що мейнфрейм може підтримувати набагато більше терміналів і дисків. Різниця між міні комп’ютером та робочою станцією полягає в тому, що робоча станція має екран, клавіатуру та мишу. Різниця між файл-сервером та робочою станцією полягає в тому, що файл-сервер має диски та периферійні пристрої, а екран, клавіатура та миша відсутні. Різниця між робочою станцією та персональним комп'ютером полягає лише в тому, що робочі станції завжди з'єднані одна з одною за допомогою локальної мережі.

У комп'ютерах різного цінового класу від робочих станцій до суперкомп'ютерів (супер серверів) використовується той самий тип мікропроцесора. Відмінності у вартості та продуктивності визначаються практично лише організацією систем пам'яті та введення/виводу (а також кількістю процесорів). Продуктивність процесорів зростає зі швидкістю 50-100% на рік. Якщо б одночасно не покращувалися характеристики систем введення/виводу, то, очевидно, розробка нових систем зайшла б у глухий кут. Важливість оцінки роботи систем вводу/виводу було усвідомлено багатьма користувачами комп'ютерів. Було розроблено спеціальні тестові програми, що дозволяють оцінити ефективність систем введення/виводу. Зокрема, такі тести застосовуються для оцінки суперкомп'ютерів, систем обробки транзакцій та файл-серверів.

***Системні та локальні шини.***

У обчислювальній системі, що складається з множини підсистем, необхідний механізм їхньої взаємодії. Ці підсистеми мають швидко та ефективно обмінюватися даними. Наприклад, процесор, з одного боку, повинен бути пов'язаний з пам'яттю, з іншого боку, необхідний зв'язок процесора з пристроями вводу/виводу. Одним із найпростіших механізмів, що дозволяють організувати взаємодію різних підсистем, є єдина центральна шина, до якої приєднуються всі підсистеми. Доступ до такої шини поділяється між усіма підсистемами.

Подібна організація має дві основні переваги: низька вартість та універсальність. Оскільки така шина є єдиним місцем приєднання для різних пристроїв, нові пристрої можуть бути легко додані, і ті самі периферійні пристрої можна навіть застосовувати в різних обчислювальних системах, що використовують однотипну шину. Вартість такої організації виходить досить низькою, оскільки для реалізації множини шляхів передачі інформації використовується єдиний набір ліній шини, що розділяється множиною пристроїв. Головним недоліком організації з єдиною шиною є те, що шина створює вузьке горло, обмежуючи, можливо, максимальну пропускну здатність вводу/виводу.

Якщо весь потік вводу/виводу повинен проходити через центральну шину, таке обмеження пропускної спроможності дуже реальне. У комерційних системах, де введення/виведення здійснюється дуже часто, а також у суперкомп'ютерах, де необхідні швидкості введення/виведення дуже високі через високу продуктивність процесора, одним із головних питань розробки є створення системи кількох шин, здатних задовольнити всі запити.

Одна з причин великих труднощів, що виникають при розробці шин, полягає в тому, що максимальна швидкість шини головним чином лімітується фізичними факторами: довжиною шини і кількістю пристроїв, що приєднуються (і, отже, навантаженням на шину). Ці фізичні обмеження неможливо довільно прискорювати шини. Вимоги швидкодії (малої затримки) системи введення/виводу та високої пропускної спроможності є суперечливими. У сучасних великих системах використовується цілий комплекс взаємопов'язаних шин, кожна з яких забезпечує спрощення взаємодії різних підсистем, високу пропускну здатність, надмірність (для збільшення стійкості до відмови) і ефективність.

Традиційно шини поділяються на шини, що забезпечують організацію зв'язку процесора з пам'яттю, та шини вводу/виводу. Шини вводу/виводу можуть мати велику довжину, підтримувати приєднання багатьох типів пристроїв, і зазвичай слідують одному з шинних стандартів. Шини процесор-пам'ять, з іншого боку, порівняно короткі, зазвичай, високошвидкісні і відповідають організації системи пам'яті для забезпечення максимальної пропускної здатності каналу пам'ять-процесор.

На етапі розробки системи, для шини процесор-пам'ять заздалегідь відомі всі типи та параметри пристроїв, які повинні з'єднуватися між собою, у той час як розробник шини вводу/виводу повинен мати справу з пристроями, що розрізняються по затримці та пропускній здатності. Як було зазначено, з метою зниження вартості деякі комп'ютери мають єдину шину для пам'яті та пристроїв вводу/виводу. Така шина часто називається системною. Персональні комп'ютери, як правило, будуються на основі однієї системної шини стандартів ISA, EISA або MCA.

Необхідність збереження балансу продуктивності зі зростанням швидкодії мікропроцесорів призвела до дворівневої організації шин у персональних комп'ютерах з урахуванням локальної шини. Локальною шиною називається шина, що електрично виходить безпосередньо на контакти мікропроцесора. Вона зазвичай поєднує процесор, пам'ять, схеми буферизації для системної шини та її контролер, а також деякі допоміжні схеми. Типовими прикладами локальних шин є VL-Bus та PCI.

***Керування введенням/виводо***м

Введення/виведення вважається однією з найскладніших областей проектування операційних систем, в якій складно застосувати загальний підхід через велику кількість приватних методів.

Складність виникає через величезну кількість пристроїв введення/виводу різноманітної природи, які мають підтримувати ОС. При цьому перед творцями ОС постає дуже непросте завдання — не лише забезпечити ефективне керування пристроями вводу/виводу, а й створити зручний та ефективний віртуальний інтерфейс пристроїв вводу/виводу, що дозволяє прикладним програмістам просто зчитувати чи зберігати дані, не звертаючи уваги на специфіку пристроїв та проблеми. розподілу пристроїв між завданнями, що виконуються.

Система введення/виводу, здатна поєднати в одній моделі широкий набір пристроїв, має бути універсальною. Вона повинна враховувати потреби існуючих пристроїв, від простої миші до клавіатур, принтерів, графічних дисплеїв, дискових накопичувачів, компакт-дисків і мереж. З іншого боку, необхідно забезпечити доступ до пристроїв введення/виводу для множини завдань, що паралельно виконуються, причому так, щоб вони якнайменше заважали один одному.

Тому найголовнішим є наступний принцип: будь-які операції з керування введенням/висновком оголошуються привілейованими і можуть виконуватися лише кодом самої ОС. Для забезпечення цього принципу у більшості процесорів навіть вводяться режими користувача та супервізора. Як правило, в режимі супервізора виконання команд введення/виводу дозволено, а в режимі користувача - заборонено. Використання команд вводу/виводу в режимі користувача викликає виняток і управління через механізм переривань передається коду ОС.

Можна назвати три основні причини, з яких не можна дозволяти кожній окремій програмі користувача звертатися до зовнішніх пристроїв безпосередньо:

1. Необхідність вирішувати можливі конфлікти доступу до пристроїв вводу/виводу.

2. Бажання збільшити ефективність використання цих ресурсів.

3. Помилки в програмах введення/виводу можуть призвести до краху всіх обчислювальних процесів, оскільки частина операцій введення/виводу здійснюється для операційної системи.

Управління введенням/виводом здійснюється операційною системою, компонентом, який найчастіше називають супервізором введення/виводу.

До переліку основних завдань, що покладаються на супервізор, входять:

• супервізор введення/виводу отримує запити на введення/виведення від прикладних завдань та від програмних модулів операційної системи.

• супервізор введення/виводу викликає відповідні розподільники каналів та контролерів, планує введення/виведення

• супервізор вводу/виводу ініціює операції вводу/виводу і у разі керування вводом/виводом з використанням переривань надає процесор диспетчеру завдань для того, щоб передати його першому завданню, що стоїть у черзі на виконання;

• при отриманні сигналів переривань від пристроїв введення/виводу супервізор ідентифікує їх та передає керування відповідною програмою обробки переривання

• супервізор введення/виводу здійснює передачу повідомлень про помилки,

• супервізор вводу/виводу посилає повідомлення про завершення операції введення/виводу процесу, що запитав цю операцію, і знімає його зі стану очікування вводу/виводу.

Якщо пристрій введення/виводу є ініціативним*, керування з боку супервізора вводу/виводу полягатиме в активізації відповідного обчислювального процесу. Ініціативним називають такий пристрій (зазвичай це датчики, зовнішній пристрій, а не пристрій введення/виведення), за сигналом переривання від якого запускається відповідна програма.

Встановивши відповідні значення параметрів у запиті на введення/виведення, що визначають необхідну операцію та кількість ресурсів, що споживаються, вони можуть передати управління супервізору введення/виводу, який і запускає необхідні логічні та фізичні операції.

Є два основних режими введення/виводу:

о режим обміну з опитуванням готовності пристрою введення/виводу та

о режим обміну з перериваннями.

Режим обміну із перериваннями за своєю суттю є режимом асинхронного керування. Для того щоб не втратити зв'язок з пристроєм (після того, як процесор видав чергову команду з управління обміном даними і переключився на виконання інших програм), може бути запущений відлік часу, протягом якого пристрій обов'язково має виконати команду і видати сигнал запиту на переривання. Максимальний інтервал часу, протягом якого пристрій

вводу/виводу або його контролер повинні видати сигнал запиту на переривання, часто називають тайм-аутом.

*Драйвери,* що працюють в режимі переривань, є складним комплексом програмних модулів і можуть мати декілька секцій: секцію запуску, одну або кілька секцій продовження і секцію завершення.

*Секція запуску* ініціює операцію вводу/виводу. Ця секція запускається увімкнення пристрою вводу/виводу чи навіть для ініціації чергової операції вводу/виводу.

*Секція продовження* (їх може бути кілька, якщо алгоритм управління обміном даними складний і потрібно кілька переривань для виконання однієї логічної операції) здійснює основну роботу з передачі даних.

*Секція завершення* зазвичай вимикає пристрій введення/виводу або завершує операцію.

Для організації використання багатьма паралельними завданнями пристроїв вводу/виводу, які не можуть бути розділяються, вводиться поняття віртуальних пристроїв. Використання принципу віртуалізації дозволяє підвищити ефективність обчислювальної системи.

Поняття віртуального пристрою ширше, ніж використання цього терміна для позначення спулінгу (SPOOLing - simultaneous peripheral operation on-line, тобто імітація роботи з пристроєм в режимі он-лайн). Головне завдання спулінгу — створити видимість паралельного поділу пристрою вводу/виводу з послідовним доступом, який фактично має використовуватися лише монопольно та бути закріпленим.

Кожна ОС має свої таблиці вводу/виводу, їх склад (кількість і призначення кожної таблиці) може відрізнятися. У деяких ОС замість таблиць створюються списки, хоча використання статичних структур даних організації вводу/виводу, зазвичай, призводить до більшому швидкодії.

Виходячи з принципу управління введенням/виводом через супервізор ОС і враховуючи, що драйвери пристроїв введення/виводу використовують механізм переривань для встановлення зворотного зв'язку центральної частини із зовнішніми пристроями, можна зробити висновок про необхідність створення принаймні трьох системних таблиць.

Перша таблиця (або список) містить інформацію про всі пристрої вводу/виводу, підключені до обчислювальної системи.

Друга таблиця призначена для реалізації ще одного принципу віртуалізації пристроїв вводу/виводу незалежності від пристрою. Призначення цієї другої таблиці - встановлення зв'язку між віртуальними (логічними) пристроями та реальними пристроями, описаними за допомогою першої таблиці обладнання.

Третя таблиця необхідна організації зворотного зв'язку між центральною частиною і пристроями вводу/виводу. Це таблиця переривань, яка вказує для кожного сигналу запиту переривання той елемент UCB, який зіставлений даному пристрою, підключеному так, що він використовує справжню лінію (сигнал) переривання.

Запит на операцію введення/виведення від програми, що виконується, надходить на супервізор. Той перевіряє системний виклик на відповідність прийнятим специфікаціям та у разі помилки повертає завданню відповідне повідомлення. Якщо ж запит коректний, він перенаправляється в супервізор вводу/виводу. Останній за логічним (віртуальним) ім'ям за допомогою таблиці DRT знаходить відповідний елемент UCB у таблиці обладнання. Якщо пристрій вже зайнятий, то описувач задачі, запит якої зараз обробляється супервізором вводу/виводу, поміщається до списку завдань, які чекають на даний пристрій. Якщо пристрій вільно, то супервізор введення/виводу визначає з UCB тип пристрою і при необхідності запускає препроцесор, що дозволяє отримати послідовність керуючих кодів і даних, яку зможе правильно зрозуміти і відпрацювати пристрій.

Таким чином, середня швидкість роботи процесора з оперативною пам'яттю на 2-3 порядки вища, ніж середня швидкість передачі даних із зовнішньої пам'яті на магнітних дисках в оперативну пам'ять.

***Засоби введення-виведення бортових систем***

Комп'ютерні системи різного призначення - промислові, інформаційно-вимірювальні, діагностичні - здійснюють зв'язок з об'єктом управління за допомогою спеціальних пристроїв зв'язку з об'єктами (УСО). До складу УСО входять датчики, виконавчі пристрої, перетворювачі рівнів та форми сигналів (аналого-цифрові АЦП, цифроаналогові перетворювачі ЦАП та ін), пристрої комутації сигналів та управління навантаженнями, а також інтерфейсні схеми для сполучення з комп'ютерною шиною.

Датчики та виконавчі пристрої знаходяться безпосередньо на об'єкті. Перетворювачі, пристрої комутації та управління навантаженнями, а також інтерфейсні схеми вбудовуються у спеціальні модулі або плати, які називаються пристроями вводу-виводу (УВВ). Іноді з метою забезпечення гнучкості та можливості роботи ПВР з різними датчиками та виконавчими пристроями перетворювачі та пристрої управління навантаженнями реалізують у вигляді окремих самостійних модулів, конструктивно винесених за межі ПВР. Пристрої введення-виведення є ключовими елементами УСО, що забезпечують інформаційну взаємодію датчиків, виконавчих пристроїв та комп'ютера.

***Класифікація пристроїв введення-виведення***

Класифікувати пристрої введення-виведення можна за різними критеріями. За особливостями застосування їх поділяють на загальнопромислові та бортові. Для останніх характерно низьке енергоспоживання, висока надійність, можливість роботи у розширеному температурному діапазоні та малі габарити.

За кількістю оброблюваних даних розрізняють недорогі (low-cost) пристрої вводу-виводу, пристрої загального призначення та спеціальні пристрої вводу-виводу (багатоканальні, високошвидкісні, мають високу роздільну здатність і працюють у граничних режимах).

На кшталт оброблюваних сигналів УВВ можна розділити на аналогові, цифрові та багатофункціональні, тобто, поєднуючи аналогові та цифрові сигнали.

За функціональним призначенням УВВ можуть бути призначені тільки для введення сигналів або тільки для виведення, або поєднувати обидві функції, а також забезпечувати прийом і передачу спеціальних сигналів (наприклад, контролери руху, обробки відео та звуку та ін.). Конструктивно УВВ є PC-плати різних форматів або окремі модулі, що реалізують прийом і передачу даних через зовнішні комп'ютерні інтерфейси.

Залежно від топології структурної схеми УВВ поділяються на дві групи: централізовані та розподілені.

Централізовані ПВР використовуються для створення нескладних автономних комп'ютерних систем керування невеликими об'єктами, тому такі ПВР іноді називають локальними. У таких системах кожен датчик чи виконавчий пристрій підключаються до ПВР комп'ютерної системи з допомогою індивідуальної лінії зв'язку, тобто. за радіальною схемою. Враховуючи високу вартість робіт з прокладання ліній зв'язку, застосування централізованих ПВР для поєднання з протяжними та складними об'єктами економічно недоцільне.

Як правило, централізовані УВВ підключаються за допомогою інтерфейсної схеми до системної шини (або однієї із зовнішніх шин) комп'ютера, тому ці УВВ розміщують в корпусі комп'ютера або поруч з ним. Зазвичай керування роботою ПВР цієї групи проводиться комп'ютером, тому наявність власних процесорів у цих ПВР не є обов'язковим, хоча сучасні ПВР нерідко містять вбудовані DSP або FPGA.

Розподілені ПВР використовуються для створення комп'ютерних систем керування складними та протяжними (сотні або тисячі метрів) у просторі об'єктами. Розподілене ПВР, структурна схема якого зображена на малюнку 8.1, реалізується на основі локальних ПВР, що об'єднуються, наприклад, за допомогою промислової мережі Fieldbus.

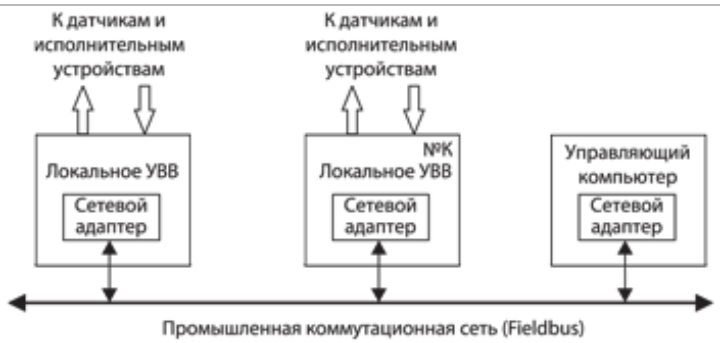



Рисунок 8.1 Структурна схема розподіленого пристрою введення-виводу

Структурні схеми локальних і централізованих УВВ багато в чому подібні, але є ряд відмінностей. У локальних ПВР розподілених систем замість шинних інтерфейсних схем застосовуються мережеві адаптери, за допомогою яких ПВП підключаються до промислової локальної мережі. Таке підключення дозволяє розташувати локальні ПВР поруч із об'єктом управління. Кожне з локальних ПВР має порівняно невелику кількість входів і виходів, до яких підключаються тільки близько розташовані датчики та виконавчі пристрої, що скорочує вартість ліній зв'язку та робіт з їх прокладання. З іншого боку, через віддаленість локальних ПВР від комп'ютера, централізоване оперативне управління їх роботою практично важко здійснити.

Крім того, кожне локальне ПВР має підтримувати той чи інший мережевий протокол обміну, прийнятий у розподіленому ПВР. Тому кожне локальне ПВР обов'язково містить вбудований процесор, що виконує функції управління ПВР та мережевої підтримки. У сучасних локальних ПВР на вбудований процесор додатково покладають деякі функції управління об'єктом, що виконуються в режимі жорсткого реального часу, що дозволяє частково розвантажити основний комп'ютер. Таким чином, сучасні розподілені ПВР трансформуються в розподілені системи управління, які в порівнянні з централізованими системами є більш продуктивними, гнучкими, легко масштабованими та надійними.

На практиці можливі випадки використання в системі одночасного двох видів ПВР: близько розташовані датчики та виконавчі пристрої підключаються за допомогою централізованих ПВР, а віддалені - за допомогою розподілених ПВР.

### *Контролери пристроїв введення-виводу*

Пристрої введення-виведення, як правило, складаються з механічних та електронних компонентів. У більшості випадків ці компоненти можна логічно розділити, щоб отримати максимально модульну та узагальнену модель.

Електронний компонент називається контролером пристрою або адаптером. У персональних комп'ютерах він має вигляд друкованої плати, вставляється в слот розширення. Механічний компонент - це сам пристрій. Ця структура представлена на рис. 8.2.

Плата контролера зазвичай забезпечується роз'ємом, до якого може бути підключений кабель, що веде до самого пристрою. Багато контролерів здатні керувати двома, чотирма або навіть вісьмома ідентичними пристроями. Якщо інтерфейс між контролером та пристроєм є стандартним, тобто визначений офіційним стандартом ANSI, IEEE чи ISO, або фактичним стандартом, це спрощує випуск окремо контролерів та пристроїв, що відповідають даному інтерфейсу.


Рисунок 8.2 Модель подключения процессора, памяти, контроллеров и устройств ввода-вывода

Ми згадуємо про різницю між контролером і пристроєм тому, що операційна система практично завжди має справу з контролером, а не з самим пристроєм. Більшість невеликих комп'ютерів взаємодія з пристроями організується за моделлю єдиної шини (див. рис. 8.2). У великих машин, мейнфреймів, застосовується інша модель з кількома шинами, які обслуговуються спеціалізованими комп'ютерами вводу-виводу, що називаються каналами введення-виводу. Така організація дозволяє зменшити навантаження на основний процесор.

Інтерфейс між пристроєм та контролером часто є інтерфейсом дуже низького рівня. Наприклад, якийсь жорсткий диск може бути відформатований по 1024 сектори на доріжку, з розміром секторів по 512 байт. Насправді з диска в контролер надходить послідовний потік бітів, що починається з заголовка сектора (преамбули), за яким слідує 4096 біт в секторі, і, нарешті, контрольна сума, також називається кодом виправлення помилок (Error-Correcting Code, ECC). Заголовок сектора записується на диск під час форматування. Він містить номери циліндрів та секторів, розмір сектора, інформацію синхронізації тощо.

Робота контролера полягає в перетворенні послідовного потоку бітів в блок байтів і корекції помилок, якщо це необхідно. Зазвичай блок байтів збирається біт за бітом у буфері контролера. Потім перевіряється контрольна сума блоку, і якщо вона збігається із зазначеною в заголовку сектора, блок належить ліченим без помилок, після чого він копіюється в оперативну пам'ять.

Контролер монітора (відеоконтролер) також працює як послідовний бітовий пристрій на такому ж низькому рівні. Він зчитує з пам'яті байти, що містять символи, які слід відобразити, і формує сигнали, що використовуються для модуляції електронної променю трубки, що змушують її виводити зображення на екран. Крім того, відеоконтролер формує сигнали, що керують горизонтальним та вертикальним переміщеннями електронного променю. На рідкокристалічному екрані ці сигнали вказують на окремі пікселі і задають їхню яскравість, імітуючи електронний промінь. Якби не контролер, програмісту довелося робити це самому. Насправді ж операційна система всього лише ініціалізує контролер, задаючи невелику кількість параметрів, таких як кількість символів або пікселів у рядку і число рядків на екрані, а основну роботу з управління розгорткою бере на себе контролер.

Контролери деяких пристроїв, особливо дисків поступово стають дуже складними. Наприклад, сучасні дискові контролери оснащені багатьма мегабайтами внутрішньої пам'яті. В результаті при виконанні операції читання контролер починає зчитувати та зберігати дані відразу після того, як головка виявляється на потрібному циліндрі (не чекаючи доступу до сектора). Таке кешування є ефективним при послідовних запитах даних. Більш того, після отримання необхідних даних контролер може продовжити кешування наступних секторів, оскільки можливість доступу до них у майбутньому велика. Подібний механізм дозволяє обслуговувати безліч запитів на читання без звернення до диска.

# ІІ ІНФОРМАЦІЙНІ СИСТЕМИ

## 9 КЛАСИФІКАЦІЯ І СТРУКТУРА ІНФОРМАЦІЙНИХ СИСТЕМ

Поняття «**інформаційні системи**», що широко використовується, практично не має єдиного концептуального визначення. Найчастіше це поняття трактується як «комплекс, що складається з інформаційного фонду та процедур: керуючої, оновлення, інформаційного пошуку та завершальної обробки, що дозволяє накопичувати, зберігати, коригувати та видавати інформацію».

Таке інтуїтивно-утилітарне визначення інформаційної системи (ІС) «випливає» і пов'язане з усталеною і звичною, проте особливою формою цілеспрямованої діяльності людини – обробкою інформації як відомостей про щось, матеріально представлених на традиційних чи машиночитаних носіях і які забезпечують ефективність розв'язання задач його основної діяльності. Тобто, «системність» тут відображає істоту функціонального відношення: склад і структура ІС визначається, виходячи з вимог до рівня ефективності обслуговування інформаційних потреб кінцевих користувачів, насамперед у частині знаходження в накопичених масивах тих записів (документів), які, ймовірно, містять потрібні відомості.

Під *інформаційною* системою розуміють сукупність засобів збору, передачі, обробки та зберігання інформації, а також персонал, який виконує такі дії (рис. 9.1).


Рисунок 9.1 Процеси в інформаційній системі

*Інформаційний фонд системи*, зазвичай, представляє базу чи сукупність БД, створюваних як табличних, ієрархічних і мережевих структур. В інтернет-ресурсах широко використовується гіпертекст, здатний представляти модель організації інформаційного простору предметної галузі. Тобто в цьому випадку не йдеться про класичні БД, а про деяку файлову, як правило, ієрархічну структуру.

У мережевих інформаційних системах використовують два способи взаємодії з кінцевими користувачами:

1. *Розподіл часу*. І тут кожен учасник мережі хіба що користується своєї ЕОМ. Основне завдання розробників та адміністраторів мережі – захист даних від несанкціонованого доступу та забезпечення взаємної ізоляції учасників.

2. *Забезпечення групових рішень* передбачає організацію взаємодії користувачів у процесі прийняття рішень. Цей метод поєднує комунікаційну, обчислювальну технологію та технологію прийняття рішень для реалізації групою осіб складних неструктурованих завдань.

Будь-яка інформаційна система повинна мати такі властивості:

● функціональністю (будь-який її об'єкт повинен містити функціонально закінчену та максимально незалежну сукупність операцій з обробки даних. Ззовні об'єкт представляє єдине ціле. Будь-яке поводження з використанням стандартизованого інтерфейсу відбувається до об'єкта в цілому);

● пов'язаністю (в об'єкті реалізується сукупність взаємозалежних функцій – методів, що працюють з тими самими даними, деякі з яких приховані для системи в цілому);

● маскуванням (для системи доступні лише ті параметри об'єкта, які складають набори вхідного та вихідного інтерфейсів об'єкта. Функції та дані, що не беруть участь у взаємодії з системою, тобто з іншими її об'єктами, приховані та поза об'єктом недоступні) та ін.

Єдина класифікація інформаційних систем ще склалася остаточно, проте, певні розробки існують (рис. 9.2). Розглянемо найважливіші аспекти класифікації ІС.

*За призначенням* функціонуючої інформації ІС поділяються на: державні, юридичні (законодавчі), ділові, фінансові, науково-технічні, навчальні, соціальні, розважальні та інші. При цьому, наприклад, фінансова інформація поділяється на: бухгалтерську, банківську, податкову та іншу, а медична (як і інші) може містити всі перелічені вище функції.

*За галузями застосування* виділяють ділову, професійну, споживчу інформацію та електронну комерцію.

*За рівнем управління* виділяють стратегічні, тактичні та оперативні інформаційні системи.

*За рівнем застосування* технічних засобів інформаційні системи ділять на автоматизовані та неавтоматизовані. При цьому автоматизовані мають на увазі автоматизацію від окремих процесів та завдань до рівня автоматизації підприємств, установ та їх сукупності в масштабах території (регіону), тобто представляють клас систем, орієнтованих на автоматизацію окремих функцій чи процесів та клас інтегрованих систем та комплексів. Ступінь автоматизації залежить від різних причин. Вона має на увазі електронну обробку та доставку даних, автоматизацію функцій та процесів управління, підтримку прийняття рішень та ін.


Рисунок 9.2 Варіант класифікації ІС

*За типами інформації*, а саме: документальної, фактографічної та документально-фактографічної пропонується виділити три типи аналогічних ІС.

*Документальні ІС* включають інформаційно-пошукові системи (ІПС), інформаційно-логічні та інформаційно-семантичні системи.

*Фактографічні ІС* діляться на дві категорії:

1) системи обробки даних (СОД);

2) автоматизовані інформаційні системи (АІС) та автоматизовані системи управління (АСУ).

Документально-фактографічні ІС містять:

1) автоматизовані документально-фактографічні інформаційно-пошукові системи науково-технічної інформації (АДФІПС НТІ),

2) автоматизовані документально-фактографічні інформаційно-пошукові системи в автоматизованій системі нормативно-методичного забезпечення управління (АДФІПС в АСНМОУ).

ІС можна класифікувати за видами оброблюваної інформації:
Текстові процесори та редактори (текст);
Графічні процесори та редактори (графіка);
Системи управління базами даних (СУБД), табличні процесори, алгоритмічні мови програмування (дані);
Експертні системи (знання), мультимедійні системи (об'єкти реального світу, що включають будь-які види інформації) та ін.

Звичайно, така класифікація досить умовна. Так, сучасний текстовий процесор може забезпечувати присутність та взаємодію практично будь-яких видів інформації, гіпертексту та можливості комунікацій. Інша річ, наскільки він задовольнятиме відповідних користувачів. Наприклад, презентаційні матеріали доцільно створювати у спеціалізованому ПЗ.

*Забезпечення інформаційних систем* поділяється на: інформаційне, технічне, математичне та програмне, методичне, лінгвістичне, правове та організаційне (Рис. 9.3).

*Інформаційне забезпечення* включає сукупність даних, методи побудови БД, а також проектних рішень щодо обсягів, розміщення, форм організації інформації, що циркулює в інформаційній системі організації.

*Технічне забезпечення* має на увазі комплекс технічних засобів, призначених для роботи інформаційної системи (ІВ), документацію на ці засоби та технологічні процеси.

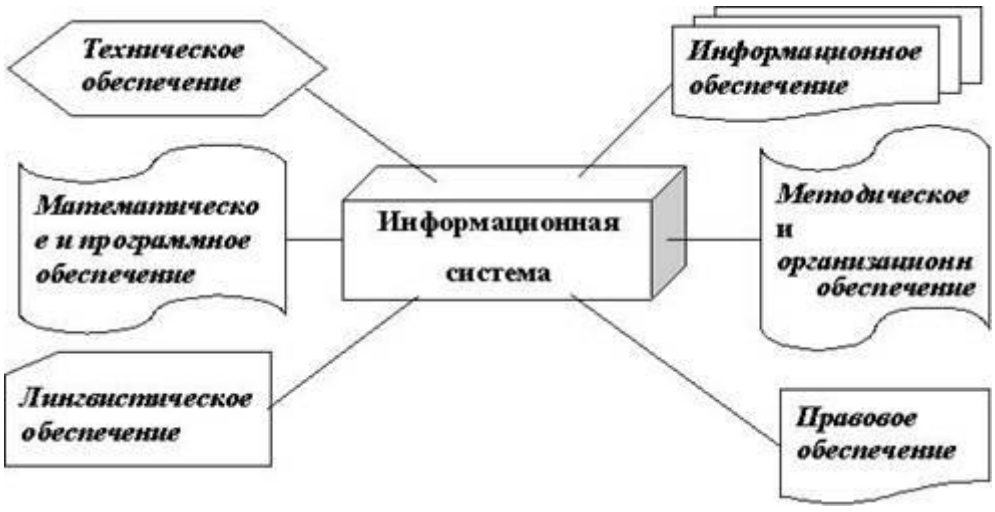



Рисунок 9.3 Підсистеми інформаційної системи

*Математичне забезпечення* представляє сукупність математичних методів, моделей, алгоритмів обробки інформації, що використовуються при вирішенні функціональних та проектних завдань у ІС.

*Програмне забезпечення* означає сукупність програм для реалізації цілей, завдань ІВ та нормального функціонування як окремих, так і комплексу технічних засобів.

*Методичне та організаційне забезпечення* складають комплекс методів, засобів та документів, що регламентують взаємодію персоналу ІВ з програмно-технічними засобами та між собою у процесі розробки та експлуатації ІВ.

*Лінгвістичне забезпечення* має на увазі сукупність мов спілкування персоналу ІС та користувачів з програмно-технічним та інформаційним забезпеченням, а також суму термінів, що використовуються в ІВ.

*Правове забезпечення* представляється правовими нормами та використовується для дотримання законності. До складу цього виду забезпечення входять закони, укази, постанови державних органів влади, накази та інструкції вищих органів та керівників організації.

Інформаційні системи - програмні системи, які вирішують прикладні завдання виходячи з використання широко відомих баз даних (БД). Інформаційні системи бувають різні за призначенням та архітектурою. Ми розглянемо, які бувають інформаційні системи, та виділимо, можливо, найскладніший вид інформаційних систем – інформаційні системи рівня корпорацій – системи, які працюють у дуже активному режимі обробки інформації; які охоплюють територією міста і навіть цілі регіони. Вони вимагають великих БД та серйозних СУБД.

Інформаційні системи бувають кількох видів. Існують класифікації інформаційних систем:

- по призначенню;
- з архітектури;
- за рівнем організацій, на яких системи використовуватимуться;
- за режимом обробки інформації;
- за обсягом баз даних, що підтримуються.

Отже, з урахуванням архітектури, що реалізується, інформаційні системи можна розбити на:

- Об'єктні або бортові (ЕОМ для пральних машин, маленькі процесори тощо);
- великі;
- системи, що базуються на технології клієнт-сервер.

З урахуванням рівня організацій, на яких системи використовуватимуться, виділяють такі інформаційні системи:

- Enterprise - wide DB (корпорації) - працюють у дуже активному режимі обробки інформації;

- Department – wide DB (рівня підрозділу) – менші обсяги інформації, кілька серверів зосереджено в одному місці (будівлі, місті);

-Workgroup DB (Рівень робочої групи) - один сервер для розробки програмної документації або програмного проекту.

За обсягом баз даних, що підтримуються, інформаційні системи діляться на:

- малі (<10 Мб);
- середні (10 Мб);
- великі (» 10 Гб);
- надвеликі (> 10 Тб).

За режимом обробки інформації інформаційні системи поділяються на:

- Системи онлайнової обробки транзакцій (наприклад, продаж квитків) - On-line Transaction Processing (OLTP). Основні транзакції – обробка інформації. Зміни у БД відбуваються не часто;

- Decision Support System. Основні операції – вибірки. Інформація має статичний характер;

- Системи аналітичної обробки інформації - On-line Analytical Processing (OLAP). Використовуються при складній математичній обробці інформації, створюються за інших систем як додаткові сервери. Такі системи використовують архіви – Data Mart (склад).

Плутанина з документами - хвороблива проблема для будь-якої компанії. Тому система автоматизації документообігу, яка дозволяє автоматизувати ручні, рутинні операції, автоматично передавати та відстежувати переміщення документів усередині корпорації, контролювати виконання доручень, пов'язаних із документами тощо. - Одна з найважливіших складових інформаційної системи.

Можна виділити два класи подібних систем:

- системи workflow;
- системи groupware.

По режиму обработки информации информационные системы делятся на:

- Системы онлайновой обработки транзакций (например, продажа билетов) - On - line Transaction Processing ( OLTP ). Основные транзакции - обработка информации. Изменения в БД происходят не очень часто;

- Decision Support System. Основные операции - выборки. Информация носит статический характер;

- Системы аналитической обработки информации - On - line Analytical Processing ( OLAP ). Используются при сложной математической обработке

информации, создаются при других системах как дополнительные сервера. Такие системы используют архивы - Data Mart (склад).

*Основне призначення* систем обох класів - автоматизація та підтримка колективної роботи в офісі, однак є деякі відмінності:

- системи класу groupware орієнтовані на автоматизацію роботи невеликого колективу і підтримують коректний поділ (тобто спільне використання) інформації групою користувачів.

- системи класу workflow орієнтовані автоматизацію роботи корпорації і підтримують поділ робіт, тобто. виконання однієї "великої" роботи групою виконавців.

Системи workflow і groupware не конкурують між собою, а скоріше доповнюють одна одну. Вибір однієї з них, а також використання в комбінації визначаються завданнями, що вирішуються корпорацією. Якщо ви дбаєте про підвищення ефективності роботи кожного окремого співробітника в колективі, перевагу варто віддати системам класу groupware (наприклад, продукт Lotus Notes (Lotus Development)).

У структуру кожної з організацій, незалежно від діяльності, входять численні підрозділи, безпосередньо здійснюють той чи інший вид діяльності компанії, і навіть дирекція, бухгалтерія, канцелярія тощо. Підрозділи компанії обмінюються між собою інформацією, а деякі з підрозділів взаємодіють із зовнішніми партнерами (див. рис.9.4).


Рисунок 9.4 Корпоративна база даних

Таким чином, будь-яка організація - це сукупність елементів, що взаємодіють (підрозділів), кожен з яких може мати свою структуру. Елементи пов'язані між собою функціонально та інформаційно, обмінюючись документами, факсами, письмовими та усними розпорядженнями. Крім того, ці елементи взаємодіють із зовнішніми системами, причому їх взаємодія також може бути як інформаційною.

Такий загальний погляд організацію дозволяє сформулювати деякі загальні принципи побудови корпоративних інформаційних систем, тобто. інформаційних систем у масштабі всієї організації.

***Етапи створення систем***

Під час створення корпоративної інформаційної системи необхідно виконати такі етапи:

- провести інформаційне обстеження організації.
- за результатами обстеження вибрати архітектуру системи та апаратно-програмні засоби її реалізації, а саме:
- систему управління корпоративною базою даних (СУБД корпоративного виду);
- систему автоматизації ділових операцій та документообігу тощо;
- проектування бази даних;
- програмна реалізація корпоративної інформаційної системи.

Інформаційна система потрібна організації для того, щоб забезпечувати інформаційно-комунікаційну підтримку її основної та допоміжної діяльності. Тому перш, ніж вести мову про структуру та функціональне наповнення інформаційної системи, необхідно розібратися з метою та завданням самої організації, щоб зрозуміти, що ж потрібно автоматизувати.

Відповіді на ці питання можна отримати лише після детального інформаційного обстеження об'єкта. Результатом обстеження є моделі діяльності компанії та її інформаційної інфраструктури, на базі яких розробляються проект корпоративної інформаційної системи, вимоги до програмно-апаратних засобів та специфікації на розробку прикладного програмного забезпечення.

Оскільки БД представляють певний набір пов'язаних даних, вони працюють завдяки програм управління базами даних (СУБД). Найпростішим способом представлення даних БД є двовимірна таблиця. Таким чином, ПЗ «Excel» є БД. Більш складною та продуктивною СУБД, що дозволяє «просунутим» користувачам самостійно або під керівництвом фахівця створювати БД, є, наприклад, ПЗ «Access». Вона використовує реляційну модель формування БД, вперше запропоновану математиком Е.Ф. Коддім у 1970 році. Для маніпулювання даними БД використовуються спеціальні

програмні мови. Найбільшого поширення нині набула мова SQL (StructuredQueryLanguage), розроблена в 1986 році Американським національним інститутом стандартів (ANSI) і прийнята в 1987 році міжнародною організацією стандартів (ISO).

*Розглянемо коротко призначення мов для роботи з БД:*

1) створення БД та таблиці з повним описом їх структури;

2) виконання основних операцій маніпулювання даними (вставка, модифікація, видалення даних та таблиць);

3) виконання простих та складних запитів, що здійснюють перетворення даних у необхідну інформацію та ін.

*Мова SQL включає дві основні компоненти*:

- мова DDL (DataDefinitionLanguage), призначена для визначення структур БД;
- мова DML (DataManipulationLanguage), яка використовується для вибірки та оновлення даних.

Сучасні способи отримання знань, розміщених у найрізноманітніших ІПС, базуються на узагальненні практичного досвіду інформаційно-аналітичної роботи, сучасних розробок в галузі ІТ, включаючи штучний інтелект. При цьому зазвичай використовується ситуаційний метод вирішення проблем – методологічна основа технології пошуку та аналізу інформації, отриманої в Інтернеті. Розглянемо типовий алгоритм пошуку інформації.

Оскільки пошук з будь-якої проблеми в Інтернеті призводить, як правило, до отримання сотень і десятків тисяч посилань на публікації – формування запитів та вибір в отриманому підмножині необхідних користувачеві матеріалів набуває інтелектуальних властивостей («datamining»), орієнтованих на інтуїтивний та формальний досвід роботи як аналітика. , і будь-якого користувача. При цьому дані та посилання на них у мережі періодично зазнають динамічних змін.

Фахівці виділяють дев'ять основних етапів виконання роботи, пов'язаних із здійсненням пошуку необхідної інформації:

· Визначення предметної галузі шуканої інформації;

· Вибір типу та джерел необхідних даних;

· Збір матеріалів для наповнення інформаційної моделі;

· Відбір найбільш корисної з них;

· Вибір методу обробки інформації: класифікація, кластеризація, регресійний аналіз тощо;

· Вибір алгоритму пошуку закономірностей;

· Пошук закономірностей, формальних правил та структурних зв'язків у зібраній інформації;

· Творча інтерпретація отриманих результатів;

· Інтеграція (об'єднання) одержаних "знань" з раніше отриманою інформацією.

В Інтернеті працюють пошукові сервери, що спеціалізуються на наданні різних видів інформації. Для цього використовуються численні пошукові інструменти. Різноманітність та конвергенція способів вилучення, візуалізації, аналізу та подання інформації – головна особливість сучасних технологій вилучення динамічно змінних даних та знань. Важливе місце у цій технології відводиться аналітикам, результати роботи яких безпосередньо залежать від їхнього вміння працювати з інформацією.

На початковій стадії створення інформаційної системи корисним може бути використання мови програмування G, яка використана в пакеті Labview, заснований на архітектурі потоків даних. Програма побудована у вигляді блочної діаграми, яка описує логіку роботи віртуального апарату, тобто програма складається з блоків подібно дитячому конструктору.

***Інформаційно-пошукові системи*** здійснюють введення, систематизацію, зберігання, видачу інформації на запит користувача без складних перетворень даних (рис. 9.5).


Рисунок 9.5 Інформаційно-пошукова система

Технологія проведення аналітиком пошуку інформації в інформаційних масивах даних:

**Вибір (збір) даних**. Цей етап передбачає отримання запиту користувача (замовника) та з'ясування (уточнення) необхідних проведення ефективного пошуку додаткових даних.

**Дослідження предметної області** полягає у формуванні у аналітика найточнішого уявлення про необхідну користувачеві інформації, а часом ще про її кількісні характеристики (обсяг окремих документів, їх сукупність та ін.).

**Аналіз інформації** – етап суміщення сформованого аналітиком уявлення про необхідну користувачеві інформації з вихідними даними, представленими користувачем. Якщо збігу немає чи існують будь-які відхилення, невідповідності тощо, сформоване аналітиком подання піддається модифікації чи перевизначенню, зокрема з уточненням у замовника необхідних аналітику додаткових даних.

**Пошук даних** включає збирання відомостей про предметну область. Для пошуку використовуються різні програмно-технічні засоби (браузери, програми пошуку інформації за принципами нечіткої логіки, нейросистеми та інших.).

**Формування рішення**. В результаті проведених аналітиком дослідження щодо постановки запиту (створення ПОЗ) та пошуку виходить, як правило, значний масив інформації, який, наприклад, у вигляді файлу або роздруківки може бути одразу (без будь-яких подальших ітерацій) переданий замовнику.

**Уточнення отриманої інформації** здійснюється у разі, коли замовником пред'явлено вимоги, що встановлюють отримання конкретних документів чи даних. І тут пошук ведеться у кількох системах (одночасно чи послідовно), їх вибираються релевантні посилання чи документи. Отримані підмасиви поєднуються в один загальний масив, і в ньому повторно проводиться пошук, але вже пертинентних даних. Подібні процедури можуть здійснюватися вручну або автоматично.

**Остаточне рішення**. На основі отриманих результатів формується остаточний, як правило, невеликий масив даних, що надаються замовнику.

# 10 СХОВИЩА ДАНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

*База даних* - набір відомостей, що зберігаються деяким упорядкованим способом чи сховище даних. Самі собою бази даних не становили б інтересу, якби не було систем управління базами даних (СУБД).

*Система управління базами даних* - це сукупність мовних та програмних засобів, яка здійснює доступ до даних, дозволяє їх створювати, змінювати та видаляти, забезпечує безпеку даних тощо. Загалом СУБД це система, що дозволяє створювати бази даних і маніпулювати відомостями з них. А здійснює цей доступ до даних СУБД у вигляді спеціальної мови – SQL.

*SQL* - мова структурованих запитів, основним завданням якого є надання простого способу зчитування та запису інформації до бази даних.

Найпростіша схема роботи з базою даних виглядає так (рис. 10.1):

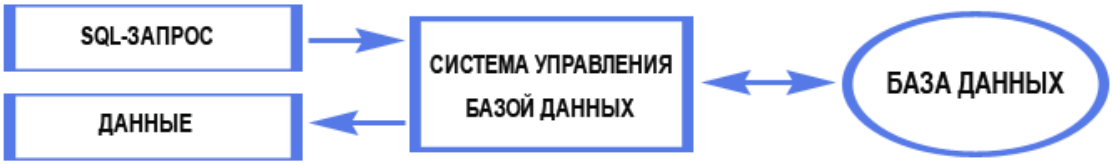



Рисунок 10.1 Структура БД

За характером використання СУБД ділять на однокористувацькі (призначені для створення та використання БД на персональному комп'ютері) і розраховані на багато користувачів (призначені для роботи з єдиною БД декількох комп'ютерів, об'єднаних у локальні мережі). Взагалі розподіл за характером використання можна подати такою схемою:

Створюючи базу даних, ми прагнемо впорядкувати інформацію за різними ознаками у тому, щоб потім витягувати з неї необхідні нам у будь-якому поєднанні. Зробити це можливо тільки якщо дані структуровані.

Структурування - це набір угод про способи представлення даних. Зрозуміло, що структурувати інформацію можна по-різному. Залежно від структури розрізняють ієрархічну, мережеву, реляційну, об'єктно-орієнтовану та гібридну моделі баз даних. Найпопулярнішою на сьогоднішній день є реляційна структура.

Ієрархічна структура бази даних

Це деревоподібна структура подання інформації. Її особливість у тому, що кожен вузол на нижчому рівні має зв'язок лише з одним вузлом на вищому рівні. Подивимося, наприклад, на фрагмент ієрархічної структури бази даних "Інститут":


Рисунок 10.2 Ієрархічна структура

Зі структури зрозуміло, що на одній кафедрі може працювати кілька викладачів. Такий зв'язок називається "*один до багатьох*" (одна кафедра – багато викладачів). Але якщо ми спробуємо додати до цієї структури групи студентів, то нам знадобиться зв'язок "*багато хто до багатьох*" (рис. 10.3):


Рисунок 10.3 Структура зв'язків

(один викладач може працювати з багатьма групами, а одна група може навчатися у багатьох викладачів), а такого зв'язку в ієрархічній структурі не може бути (т.к. зв'язок може бути тільки з одним вузлом на вищому рівні). Це основний недолік такої структури бази даних.

***Мережева структура бази даних***

По суті це розширення ієрархічної структури. Все те саме, але існує зв'язок "багато до багатьох". Мережева структура бази даних дозволяє нам додати групи до нашого прикладу. Недоліком мережевої моделі є складність розробки серйозних програм.

***Реляційна структура бази даних***

Всі дані представлені у вигляді простих таблиць, розбитих на рядки та стовпці, на перетині яких розташовані дані.

***Об'єктно-орієнтовані та гібридні бази даних***

У об'єктно-орієнтованих базах даних дані зберігаються як об'єктів, що дуже зручно. Але нині такі БД ще поширені, т.к. поступаються у продуктивності реляційним.

***Гібридні БД*** поєднують у собі можливості реляційних та об'єктно-орієнтованих, тому їх часто називають об'єктно-реляційними. Прикладом такої СУБД є Oracle, починаючи з восьмої версії (рис. 10.4).


Рисунок 10.4 Структура СУБД

Сьогодні кількість використовуваних СУБД обчислюється десятками. Найбільш відомі однокористувацькі СУБД - Microsoft Visual FoxPro і Access, розраховані на багато користувачів - MS SQL Server, Oracle і MySQL.

Реляційні бази даних складаються з таблиць. Кожна таблиця складається зі стовпців (їх називають полями чи атрибутами) та рядків (їх називають записами чи кортежами). Таблиці в реляційних базах даних мають ряд властивостей. Основними є такі:

• У таблиці не може бути двох однакових рядків. У математиці таблиці, що мають таку властивість, називають відносинами - англійською relation, звідси і назва - реляційні.

• Стовпці розміщуються у певному порядку, який створюється під час створення таблиці. У таблиці може бути жодного рядка, але обов'язково має бути хоча б один стовпець.

• Кожен стовпчик має унікальне ім'я (у межах таблиці), і всі значення в одному стовпці мають один тип (число, текст, дата...).

• На перетині кожного стовпця та рядка може знаходитись лише атомарне значення (одне значення, що не складається з групи значень).

Таблиці, що задовольняють цю умову, називають нормалізованими.

**Приклад.** Припустимо, ми захотіли створити базу даних для форуму. У форумі є зареєстровані користувачі, які створюють теми та залишають повідомлення на цих темах. Ця інформація повинна зберігатися в базі даних.

Теоретично (на папері) ми можемо все це розмістити в одній таблиці, наприклад, так:

| Имя | E-mail | Пароль | Созданные темы | Созданные сообщения |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |

Але це суперечить властивості атомарності (одне значення в одному осередку), а в стовпцях Теми та Повідомлення у нас передбачається необмежену кількість значень. Отже, нашу таблицю треба розбити на три: Користувачі, Теми та Повідомлення.

**Пользователи**

| Имя | E-mail | Пароль |
|---|---|---|
| | | |
| | | |
| | | |

**Темы**

| Наименование | Автор |
|---|---|
| | |
| | |
| | |

**Сообщения**

| Текст | Автор |
|---|---|
| | |
| | |
| | |

Наша таблиця Користувачі задовольняють всі умови. А ось таблиці Теми та Повідомлення – ні. Адже в таблиці не може бути двох однакових рядків, а де гарантія, що один користувач не залишить два однакових повідомлення, наприклад:

**Сообщения**

| Текст | Автор |
|---|---|
| Думаю надо сделать так… | Кирилл |
| Согласен | Вася |
| А еще можно сделать так… | Семен |
| Согласен | Вася |

Крім того, ми знаємо, що кожне повідомлення обов'язково стосується будь-якої теми. А як це можна дізнатися із наших таблиць? Ніяк. Для вирішення цих проблем у реляційних базах даних існують ключі.

*Первинний ключ* (скорочено PK – primary key) – стовпець, значення якого у всіх рядках різні. Первинні ключі можуть бути логічними (природними) та сурогатними (штучними). Так, для нашої таблиці Користувачі первинним ключем може стати стовпець e-mail (адже теоретично може бути двох користувачів з однаковим e-mail). Насправді краще використовувати сурогатні ключі, т.к. їхнє застосування дозволяє абстрагувати ключі від реальних даних. Крім того, первинні ключі міняти не можна, а якщо у користувача зміниться e-mail?

Сурогатний ключ є додатковим полем у базі даних. Як правило, це порядковий номер запису (хоча ви можете задавати їх на власний розсуд, контролюючи, щоб вони були унікальні). Давайте внесемо поля первинних ключів до наших таблиць:

**Пользователи**

| id пользователя | Имя | E-mail | Пароль |
|---|---|---|---|
| 1 | Кирилл | kirill@mail.ru | Gh345fgh |
| 2 | Вася | vasy@rambler.ru | As3bh7 |
| 3 | Семен | semen@yandex.ru | gk4bb6 |

**Темы**

| id темы | Наименование | Автор |
|---|---|---|
| 1 | О рыбалке | Кирилл |
| 2 | Велосипеды | Вася |
| 3 | Ночные клубы | Семен |
| 4 | О рыбалке | Вася |

**Сообщения**

| id сообщения | Текст | Автор |
|---|---|---|
| 1 | Думаю надо сделать так… | Кирилл |
| 2 | Согласен | Вася |
| 3 | А еще можно сделать так… | Семен |
| 4 | Согласен | Вася |

Тепер кожен запис у наших таблицях є унікальним. Нам залишилося встановити відповідність між темами та повідомленнями у них. Робиться це також за допомогою первинних ключів. У таблицю повідомлення ми додамо ще одне поле:

**Сообщения**

| id сообщения | Текст | Автор | id темы |
|---|---|---|---|
| 1 | Думаю надо сделать так… | Кирилл | 1 |
| 2 | Согласен | Вася | 4 |
| 3 | А еще можно сделать так… | Семен | 1 |
| 4 | Согласен | Вася | 1 |

Тепер зрозуміло, що повідомлення з id=2 належить темі "Про рибалку" (id теми = 4), створеної Васею, а інші повідомлення належать темі "Про рибалку" (id теми = 1), створеної Кирилом. Таке поле називається зовнішній ключ (скорочено FK – foreign key). Кожне значення цього поля відповідає первинному ключу з таблиці "Теми". Так встановлюється однозначна відповідність між повідомленнями та темами, до яких вони належать. Останній аспект. Припустимо, у нас додався новий користувач, і звуть його теж Вася:

**Пользователи**

| id пользователя | Имя | E-mail | Пароль |
|---|---|---|---|
| 1 | Кирилл | kirill@mail.ru | ******* |
| 2 | Вася | vasy@rambler.ru | ******* |
| 3 | Семен | semen@yandex.ru | ******* |
| 4 | Вася | vasy@mail.ru | ******* |

Як ми дізнаємось, який саме Вася залишив повідомлення? Для цього поля автор у таблицях "Теми" та "Повідомлення" ми зробимо також зовнішніми ключами:

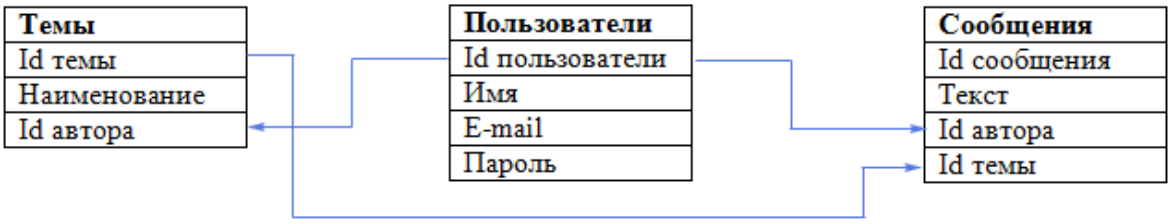
**Темы**

| id темы | Наименование | id автора |
|---|---|---|
| 1 | О рыбалке | 1 |
| 2 | Велосипеды | 2 |
| 3 | Ночные клубы | 3 |
| 4 | О рыбалке | 1 |
| 5 | К кому обратиться | 4 |

**Сообщения**

| id сообщения | Текст | id автора | id темы |
|---|---|---|---|
| 1 | Думаю надо сделать так… | 1 | 1 |
| 2 | Согласен | 2 | 4 |
| 3 | А еще можно сделать так… | 3 | 1 |
| 4 | Согласен | 2 | 1 |

Наша база даних готова. Схематично її можна так:




У нашій маленькій базі даних лише три таблички, а якби їх було 10 чи 100? Зрозуміло, що відразу неможливо уявити всі таблиці, поля та зв'язки, які нам можуть знадобитися. Саме тому проектування бази даних починається із її концептуальної моделі.

Концептуальна модель — це відображення предметної галузі, на яку розробляється база даних. Зазначимо, що це певна діаграма з прийнятими позначеннями елементів. Так, всі об'єкти, що позначають речі, позначаються як прямокутника. Атрибути, що характеризують об'єкт - як овалу, а зв'язок між об'єктами - ромбами. Потужність зв'язку позначаються стрілками (у напрямку, де потужність дорівнює багатьом - подвійна стрілка, а з боку, де вона дорівнює одиниці - одинарна).

Як приклад розглянемо інтернет-магазин. У магазину є товари, які постачаються постачальниками та купуються покупці. Це можна уявити трьома об'єктами та двома зв'язками (рис. 10.5):

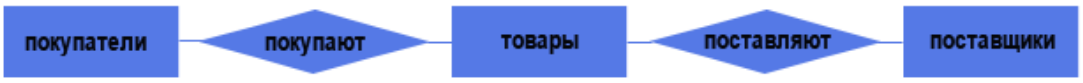


Рисунок 10.5 Структура руху товарів

Але як постачальник постачає товари? Він робить постачання, яке підтверджується документом. Аналогічно і покупець робить покупку, яка також може підтверджуватись документом. Таким чином, постачання та купівля можуть розглядатися як самостійні об'єкти (рис. 10.6):

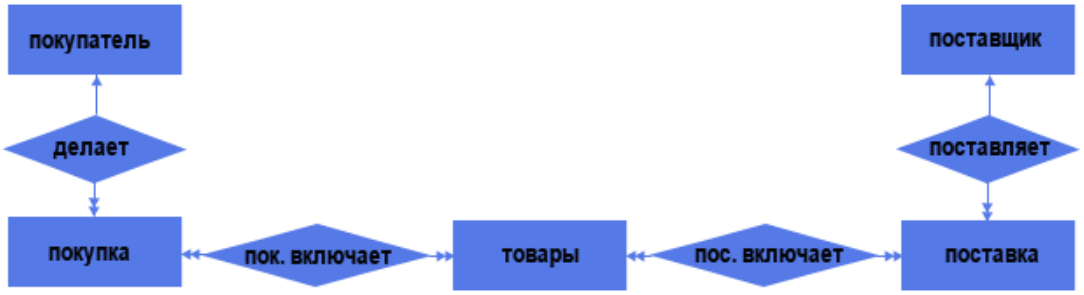



Рисунок 10.6 Об'єктне уявлення

Тепер у нас п'ять об'єктів та чотири зв'язки. Два зв'язку "один до багатьох" (один постачальник може здійснити кілька поставок, але кожне постачання здійснюється тільки одним постачальником, аналогічно і для зв'язку Покупець - Купівля) і два зв'язку "багато до багатьох" (кожне постачання може містити кілька товарів, а один і той самий товар може утримуватися у кількох поставках, аналогічно й у зв'язку Покупка - Товар).

Але зв'язки "багато хто до багатьох" неприпустимі в реляційної моделі, тому кожен такий зв'язок треба замінити на два зв'язку "один до багатьох". Робиться це додаванням проміжного об'єкта (рис. 10.7):

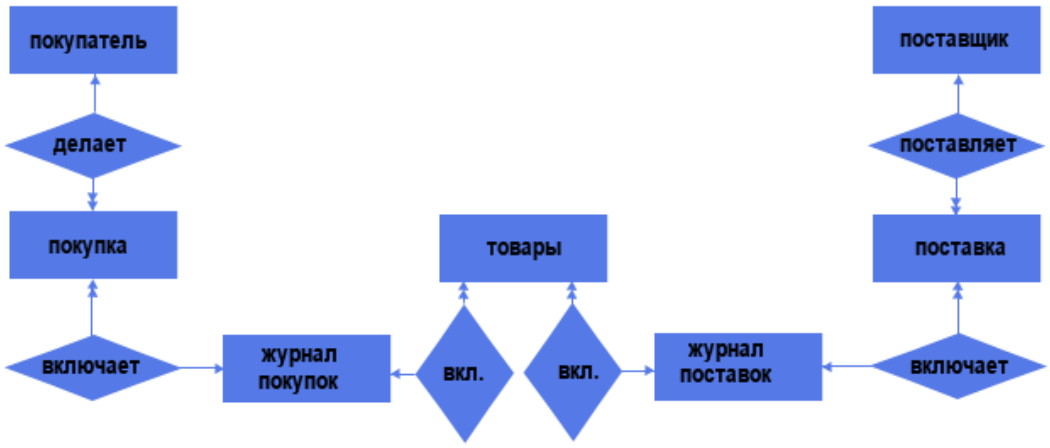



Рисунок 10.7 Модернізація структури

Таким чином, у нас з'явилося ще два об'єкти - журнал покупок і журнал постачання, зі зв'язками "один до багатьох" (один журнал постачання може включати кілька постачань, але кожна постачання може входити тільки в один журнал, аналогічно і для інших).

Кожен об'єкт нашого магазину має свої атрибути (рис. 10.8):


Рисунок 10.8 Концептуальна схема БД

Взагалі, якщо предметна область велика, її корисно розбити кілька локальних предметних областей (наша концептуальна модель відбиває саме локальну предметну область). Об'єм локальної області вибирається таким чином, щоб до неї входило трохи більше 6-7 об'єктів. Після створення моделей кожної виділеної предметної області проводиться об'єднання локальних концептуальних моделей одну загальну, зазвичай, досить складну схему.

Обмежимося створеною моделлю. Тепер нашу концептуальну модель треба перетворити на реляційну модель даних, тобто. таблиці, поля, ключі і т.д. Отже, треба побудувати набір таблиць. Зробити це неважко, т.к. таблиці – це наші об'єкти, а поля таблиць – атрибути об'єктів. Набір попередніх таблиць, виходячи з нашої концептуальної моделі, має такий вигляд:

| **Покупатель** |
|---|
| Id покупателя (PK) |
| ФИО |
| E-mail |

| **Поставщик** |
|---|
| Id поставщика (PK) |
| Наименование |
| Адрес |

| **Покупка** |
|---|
| Id покупки (PK) |
| Id покупателя (FK) |
| Дата |

| **Поставка** |
|---|
| Id поставки (PK) |
| Id поставщика (FK) |
| Дата |

| **Товар** |
|---|
| Id товара (PK) |
| Наименование |
| Цена |

| **Журнал покупок** | |
|---|---|
| Id покупки (FK) | } PK |
| Id товара (FK) | |
| Количество | |

| **Журнал поставок** | |
|---|---|
| Id поставки (FK) | } PK |
| Id товара (FK) | |
| Количество | |

Таким чином, у нас визначено таблиці, поля, первинні ключі (PK) та зв'язки (FK). Зверніть увагу, у таблицях Журнал постачання та Журнал покупок первинні ключі - складові, тобто. складаються із двох полів. Теоретично бувають таблиці, де всі поля є одним складовим ключем.

Переходимо до другого пункту, саме до нормалізації відносин (таблиць). Нормалізація - це покроковий, оборотний процес заміни вихідної схеми іншою схемою, де таблиці мають простішу і логічну структуру. Для чого це потрібно?

По-перше, для усунення надмірності даних. Наприклад, у нашому прикладі для форуму (з третього уроку) ми залишили б ось таку таблицю:

| Id сообщения | Тема | Текст | Автор |
|---|---|---|---|
| 1 | О рыбалке | Думаю надо сделать так… | Кирилл |
| 2 | О рыбалке | Согласен | Вася |
| 3 | О рыбалке | А еще можно сделать так… | Семен |
| 4 | Велосипеды | Согласен | Вася |

У полі Теми часто повторюються ті самі назви. Крім того, що для їх зберігання будуть потрібні додаткові ресурси пам'яті, при дублюванні інформації дуже неважко припуститися помилки при введенні значень атрибуту, внаслідок чого БД перейде в неузгоджений стан.

Крім того, під час роботи з такими таблицями можуть виникнути так звані аномалії оновлення. Наприклад, якщо ми видалимо з цієї таблиці четверте повідомлення, то разом з ним зникне інформація про тему. Така ситуація є аномалією видалення. Якщо ми вирішимо змінити назву теми, нам доведеться переглянути всі рядки і в кожному замінити стару тему на нову. Це так звана аномалія модифікації. Існують інші види аномалій.

Не завжди ці недоліки можна врахувати відразу. Для їхнього усунення і застосовується процес нормалізації. Він включає низку правил, що використовуються для перевірки всіх таблиць бази даних. Розрізняють:

- 1НФ – перша нормальна форма
- 2НФ – друга нормальна форма
- 3НФ – третя нормальна форма
- НФБК – нормальна форма Бойса-Кодда
- 4НФ – четверта нормальна форма
- 5НФ – п'ята нормальна форма.

Кожна нормальна форма накладає певні обмеження даних. Кожна нормальна форма вищого рівня передбачає, що аналізована таблиця вже перебуває у нормальній формі рівень нижче аналізованої. У ході нормалізації схема бази даних стає все суворішою, а її таблиці все менш схильні до різного роду аномалій.

Для реляційних баз даних необхідно, щоб її таблиці перебували у 1НФ. Нормальні форми вищих рівнів можуть використовуватися розробниками на власний розсуд. Проте грамотний фахівець прагне довести рівень нормалізації бази даних хоча б до 3НФ, виключивши надмірність даних і аномалії оновлення. Треба сказати, що НФБК, 4НФ та 5НФ використовуються вкрай рідко. Тому й ми розглянемо лише перші три.

***Перша нормальна форма***

Таблиця знаходиться у першій нормальній формі, якщо всі її поля мають прості (атомарні) значення. Саме поняття атомарності визначити досить складно. Значення атомарне в одному випадку може бути неатомарним в іншому. Загальний принцип тут такий: значення не атомарне, якщо воно використовується частинами. Зрозуміліше буде з прикладу.

У нашій таблиці Постачальники мають поле Адреса. Якщо наш магазин працює тільки з постачальниками з одного міста, значення поля Адреса можна вважати атомарними, а саму таблицю - наведеною до 1НФ.

Але якщо наші постачальники знаходяться в різних містах? Тоді, посилаючи машину за товарами до певного міста, ми маємо бути впевнені, що вона забере товари у всіх постачальників, які перебувають у цьому місті. Тобто. нам можуть знадобитися відомості про постачальників, які знаходяться у певному місті. У цьому випадку значення в полі Адреса вже не є атомарними (т.к. ми використовуємо частину адреси), і для приведення таблиці до 1НФ нам треба виділити ще одне поле.

| **Поставщик** |
|---|
| Id поставщика (PK) |
| Наименование |
| Город |
| Адрес |

Таким чином, треба проаналізувати всі таблиці нашої бази даних. Так, у таблиці Покупець є поле ПІБ. Якщо ми збираємося, наприклад, вітати наших покупців з іменинами (які, як відомо, залежать від імені), то це поле довелося б розбити на три: Прізвище, Ім'я та По-батькові. Наш магазин цього робити не збирається, тому поле ПІБ можна вважати атомарним, а таблицю – наведеною до 1НФ. Для запитів нашого магазину всі інші таблиці наведено до 1НФ.

***Друга нормальна форма***

Ця форма застосовується до таблиць із складовими ключами. Таблиця, у якій первинний ключ включає лише одне поле, завжди знаходиться у 2НФ.
Таблиця знаходиться у другій нормальній формі, якщо вона знаходиться у першій нормальній формі, а кожне неключове поле функціонально повно залежить від складеного ключа.

У нашій базі даних дві таблиці мають складовий ключ - Журнал покупок та Журнал постачання. Значення поля Кількість залежить як від Поставки (Покупки), так і від Товару. Отже, наші таблиці перебувають у 2НФ.
Але припустимо, що на етапі концептуального моделювання нашої бази даних ми не виділили об'єкти Постачання та Покупка. Тоді наші таблиці могли б мати такий вигляд:


Подивимося тепер на таблицю Журнал постачання: поле Кількість залежить від Найменування товару та від Дати поставки, але не залежить від того, хто поставив товар (поле Постачальника). Тобто. таблиця перебуває у 2НФ. Якби на етапі концептуального моделювання нашої бази даних ми не виділили об'єкти Постачання та Покупка, нам довелося б це робити зараз. Але ми їх виділили, тому всі наші таблиці знаходяться у 2НФ.

***Третя нормальна форма***

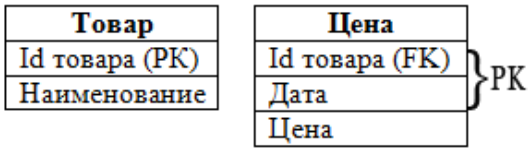
Таблиця знаходиться у третій нормальній формі, якщо вона знаходиться у другій нормальній формі, і кожне неключове поле нетранзитивно залежить від первинного ключа.

Транзитивна залежність спостерігається в тому випадку, якщо одне із двох неключових полів залежить від первинного ключа, а інше залежить від першого неключового поля. На прикладі буде зрозуміліше.

Подивимося нашу таблицю Товар. У ній є поле Ціна, але ціни, як відомо, мають властивість змінюватись. Якщо ми їх змінюватимемо прямо тут, то пропадатиме вся інформація про попередні ціни. Щоб не втрачати цю інформацію, треба додати поле Дата (якщо змінилася ціна). Тоді наша таблиця виглядатиме так:

| Товар |
|---|
| Id товара (PK) |
| Наименование |
| Дата |
| Цена |

Навіть не вдаючись до 3НФ видно, що така таблиця міститиме надмірну інформацію. Але подивимося на її поля: поля Найменування та Дата залежать від ID товару, а поле Ціна залежить також і від Дати. Тобто. таблиця не знаходиться у 3НФ. Для усунення транзитивної залежності необхідно провести розщеплення об'єкта на два:




Решта таблиці нашої бази даних перебувають у 3НФ. До речі, у таблиці Товар можна було й не вводити поле id товару, а зробити первинним ключем поле Найменування, але як говорилося у третьому уроці сурогатні ключі все-таки краще.

Підведемо підсумок. Схема нашої бази даних після нормалізації дещо змінилася і виглядає так:


Таким чином, ми перетворили нашу концептуальну модель на реляційну. Далі необхідно цю модель реалізувати у конкретній СУБД. Для цього нам знадобиться сама СУБД та знання мови SQL.

Проектування БД процес, як правило, трудомісткий та нешвидкий. Адже треба дуже добре вивчити предметну область, щоб врахувати всі нюанси, побажання та вимоги користувачів. Потім всю зібрану інформацію зобразити як об'єктів, атрибутів і зв'язків. Причому зробити це треба раціональніше. Так, база даних - це лише сховище даних, але від того наскільки грамотно ви організуєте це сховище, залежатиме робота вашого додатка, що використовує дані.

Інформаційна система може бути просторово розподілена, мати значні відстані між підсистемами, деякі з яких можуть бути встановлені на рухомих об'єктах, що суттєво обмежує динамічні характеристики системи за рахунок обмеженої пропускної спроможності каналів зв'язку та черг обслуговування. Для вирішення цієї проблеми запропоновано розподілені бази даних (рис. 10.9).

Розподілена база даних означає базу даних, яка не обмежена однією системою, але розподілена по кількох сайтах, тобто на кількох серверах або через мережу серверів у кількох місцях. Це база даних, що складається із двох або більше файлів. Крім того, розподілена система баз даних знаходиться на кількох сайтах, які не мають спільних фізичних компонентів. Це може бути необхідно, якщо різні користувачі по всьому світу потребують доступу до певної бази даних або в одній мережі, або в різних мережах. Ми зберігаємо частини бази даних у кількох фізичних місцях, і навіть розподіляємо обробку

між кількома серверами баз даних. Ми повинні керувати базою даних таким чином, щоб вона була схожа на єдину для користувачів.


Рисунок 10.9 Зосереджена та розподілена БД

***Архітектура розподіленої бази даних***

Розподілені бази даних можуть бути однорідними чи гетерогенними.

*Однорідна база даних*

Однорідні розподілені системи баз даних видаються користувачеві як єдина система, і їх набагато простіше проектувати та керувати ними. Однак, щоб розподілена система баз даних була однорідною, структури даних у кожному місці мають бути ідентичними чи сумісними. Додаток бази даних, який ми використовуємо в кожному місці, також має бути ідентичним або сумісним. В однорідній базі даних всі різні сайти зберігають дані однаково. Це операційна система, СУБД і структури даних, що використовуються, також однакові на всіх сайтах. Це полегшує управління ними.

*Неоднорідна база даних*

У гетерогенній розподіленій базі даних обладнання, операційні системи або програми бази даних можуть бути різними в кожному місці. Різні сайти можуть використовувати різні моделі даних та програмне забезпечення. Різниця в модель даних може обробляти, проте запити ускладнюють. Це може спричинити проблеми при обробці транзакцій. Сайт також може бути несумісний або зовсім незнайомий з іншими сайтами. Користувачі в одному місці можуть читати дані в іншому, але не можуть завантажувати або змінювати

їх. Щоб вирішити ці проблеми, гетерогенна база даних потребує перекладів для взаємодії між різними сайтами. Гетерогенні розподілені бази даних часто важко використовувати, що робить їх економічно невигідними для багатьох компаній.

***Розподіл сховищ даних***

Існують два способи зберігання даних на різних сайтах. Це:

***Копіювання.***

Реплікація бази даних гарантує, що дані у розподілених базах даних будуть оновлені. При реплікації всі дані зберігаються надмірно на 2 або більше сайтах. Якщо повна база даних зберігається на всіх сайтах, ми говоримо про повністю надмірну базу даних. Під час реплікації системи фактично зберігають копії даних.

Репліковані дані можна розділити на дві категорії: дані лише для читання та дані з можливістю запису. Версії реплікованих даних лише для читання допускають редагування лише на першому вказаному сервері сайту. Потім дані коригуються інших наступних серверах. Дані, що записуються, можуть бути змінені, але перший сервер сайту змінюється негайно.

Великою перевагою цього є висока доступність даних у різних місцях. У такий спосіб можна обробляти запити паралельно.

Проте є недоліки. Дані повинні постійно оновлюватись. Будь-які зміни, внесені на один сайт, мають бути перенесені на решту, інакше це призведе до невідповідностей. Це створює велике навантаження на системи та мережу. Крім того, перевірка паралелізму стає набагато складнішою, оскільки блокування записів має виконуватися на всіх сайтах одночасно.

***Fragmentatie***

За такого підходу дані фрагментовані, тобто діляться більш дрібні частини. Фрагменти розповсюджуються та зберігаються на різних сайтах, де вони потрібні. Розподіл фрагментів здійснюється розумно, щоб їх можна було використовувати для відновлення вихідних даних, якщо один із сайтів вийде з ладу. Таким чином, немає втрати даних.

Перевага фрагментації в тому, що нам не потрібно робити копії даних, доки база даних залишається узгодженою.

Тому в деяких випадках використовується гібридний підхід фрагментації та реплікації.

***Система управління розподіленою базою даних (DDBMS)***

Централізована система управління розподіленою базою даних логічно поєднує дані. Такий користувач відчуває, що всі дані зберігаються на одному сайті (з ним). DDBMS - це одна СУБД, яка синхронізує всі дані безпосередньо або періодично і гарантує, що всі записи, оновлення та видалення, які виконуються на сайті, автоматично виконуються на даних, що зберігаються в іншому місці.

Однак централізована база даних має лише один файл бази даних, розташований на одному сайті та доступний через єдину мережу.

***Особливості розподілених баз даних***

Коли набір розподілених баз даних логічно переплітається, вони утворюють практично єдину логічну базу даних.

• Сервери на кожному сайті пов'язані через мережу, але не мають багатофункціональних функцій.

• Операція не залежить від обладнання.

• Розподілена база даних фізично зберігає дані на кількох сайтах та керує ними незалежно.

• Операція не залежить від операційної системи.

• Операція не залежить від мережі.

• Нарешті операція не залежить від СУБД.

Як правило, розподілені бази даних включають такі функції:

• Можливості працювати самостійно.

• Розподілена обробка запитів.

• Обробка розподілених транзакцій.

• Транзакційна прозорість.

***Переваги розподіленої бази даних***

Використання розподіленої бази даних дає безліч переваг.

• Якщо в централізованій базі даних відбуваються збої, система повністю зупиняється. Однак у разі відмови компонента в системі розподіленої бази даних система продовжить роботу зі зниженою продуктивністю, доки помилка не буде усунена.

• Системи розподілених баз даних дозволяють нам знизити витрати на підключення, розміщуючи дані ближче до сайту, який їх найбільше використовує. Це неможливо у централізованій системі.

• Ми можемо розробляти розподілені бази даних за модульним принципом. Це означає, що ми можемо розширювати системи, додаючи нові сервери та локальні дані на новий сайт та підключаючи їх до розподіленої системи.

# 11 ЛОКАЛЬНІ ТА ГЛОБАЛЬНІ МЕРЕЖІ, МЕРЕЖЕВІ ТЕХНОЛОГІЇ ОБРОБКИ ДАНИХ

При фізичному з'єднанні двох або більше комп'ютерів утворюється **комп'ютерна мережа**. У разі, для створення комп'ютерних мереж необхідно спеціальне апаратне забезпечення - мережне обладнання та спеціальне програмне забезпечення - мережні програмні засоби.

Вже зараз є сфери людської діяльності, які принципово не можуть існувати без мереж (наприклад, робота банків, великих бібліотек тощо). Мережі також використовуються при керуванні великими автоматизованими виробництвами, газопроводами, електростанціями тощо, фізичні канали, які зазвичай називають **середою передачі.**

Призначення всіх видів комп'ютерних мереж визначається двома функціями:

· Забезпечення спільного використання апаратних та програмних ресурсів мережі;

· Забезпечення спільного доступу до ресурсів даних.

Наприклад, всі учасники локальної мережі можуть спільно використовувати один спільний пристрій друку – мережевий принтер або, наприклад, ресурси жорстких дисків одного виділеного комп'ютера – файлового сервера. Аналогічно можна спільно використовувати програмне забезпечення. Якщо у мережі є спеціальний комп'ютер, виділений для спільного використання учасниками мережі, він називається **файловим сервером**.

Групи співробітників, які працюють над одним проектом у межах локальної мережі, називаються **робочими групами**. У межах однієї локальної мережі можуть працювати кілька робочих груп. Учасники робочих груп можуть мати різні права для доступу до загальних ресурсів мережі. Сукупність прийомів поділу та обмеження прав учасників комп'ютерної мережі називається **політикою мережі**. Керування мережевими політиками називається адмініструванням мережі. Особа, яка управляє організацією роботи учасників локальної комп'ютерної мережі, називається **системним адміністратором**.

### *Основні характеристики та класифікація комп'ютерних мереж*

**За територіальною поширеністю** мережі можуть бути локальними, глобальними та регіональними (рис. 11.1).

Рисунок 11.1 Комп'ютерні мережі

**Локальна мережа** (LAN – Local Area Network) – мережа в межах підприємства, установи, однієї організації.
**Регіональна мережа** (MAN – Metropolitan Area Network) – мережа в межах міста чи області.
**Глобальна мережа** (WAN – Wide Area Network) – мережа на території держави або групи держав.
**За швидкістю передачі інформації** комп'ютерні мережі умовно поділяються на низько-, середньо- та високошвидкісні:
· **низькошвидкісні мережі** – до 10 Мбіт/с;
· **середньошвидкісні мережі** – до 100 Мбіт/с;
· **високошвидкісні мережі** - понад 100 Мбіт/с.
За типом середовища передачі мережі поділяються на:
· **дротові** (на коаксіальному кабелі, на кручений парі, оптоволоконні);
· **бездротові** з передачею інформації по радіоканалах або в інфрачервоному діапазоні.

За способом організації взаємодії комп'ютерів мережі ділять на **однорангові** та з **виділеним сервером** (**ієрархічні** мережі).
Усі комп'ютери однорангової мережі рівноправні. Будь-який користувач мережі може отримати доступ до даних, які зберігаються на будь-якому комп'ютері.
Головна перевага однорангових мереж – це простота встановлення та експлуатації. Головний недолік полягає в тому, що в умовах однорангових мереж утруднено вирішення питань захисту інформації. Тому такий спосіб організації мережі використовується для мереж з невеликою кількістю комп'ютерів і там, де питання захисту не є принциповим.
В ієрархічній мережі при встановленні мережі заздалегідь виділяються один або кілька **серверів** - комп'ютерів, що управляють обміном даних по мережі та розподілом ресурсів. Будь-який комп'ютер, який має доступ до послуг сервера, називають **клієнтом мережі** або **робочою станцією**.

Сервер в ієрархічних мережах - це постійне сховище ресурсів, що розділяються. Сам сервер може бути клієнтом лише сервера вищого рівня ієрархії. Сервери зазвичай являють собою високопродуктивні комп'ютери, можливо, з кількома процесорами, що працюють паралельно, вінчестерами великої ємності і високошвидкісною мережевою картою.
Ієрархічна модель мережі є найкращою, оскільки дозволяє створити найбільш стійку структуру мережі і більш раціонально розподілити ресурси. Також перевагою ієрархічної мережі є вищий рівень захисту даних. До недоліків ієрархічної мережі, порівняно з одноранговими мережами, належать:

1. Необхідність додаткової ОС для сервера.
2. Вища складність установки та модернізації мережі.

3. Необхідність виділення окремого комп'ютера як сервер.

**За технологією використання сервера** розрізняють мережі з архітектурою **файл-сервер** та мережі з архітектурою **клієнт-сервер**. У першій моделі використовується файловий сервер, на якому зберігається більшість програм та даних. На вимогу користувача йому надсилаються необхідна програма та дані. Обробка інформації виконується на робочій станції.

У системах з архітектурою клієнт-сервер обмін даними здійснюється між додатком-клієнтом та додатком-сервером. Зберігання даних та їх обробка провадиться на потужному сервері, який виконує також контроль за доступом до ресурсів та даних. Робоча станція отримує лише результати запиту.

До основних характеристик мереж відносяться:

**Пропускна здатність** – максимальний обсяг даних, що передаються мережею за одиницю часу. Пропускна здатність вимірюється в Мбіт/с.

**Час реакції мережі** - час, що витрачається програмним забезпеченням та пристроями мережі на підготовку до передачі інформації цим каналом. Час реакції мережі вимірюється мілісекундами.

**Топології мереж**

Топологією мережі називається фізичну чи електричну конфігурацію кабельної системи та з'єднань мережі. У топології мереж застосовують кілька спеціалізованих термінів:

· Вузол мережі - комп'ютер, або комутуючий пристрій мережі;

· Гілка мережі - шлях, що з'єднує два суміжні вузли;

· Кінцевий вузол - вузол, розташований в кінці лише однієї гілки;

· Проміжний вузол - вузол, розташований на кінцях більш ніж однієї гілки;

· суміжні вузли - вузли, з'єднані принаймні одним шляхом, що не містить жодних інших вузлів.

Існує всього 5 основних типів топології мереж:

1. Топологія "Загальна Шина". У цьому випадку підключення та обмін даними здійснюється через загальний канал зв'язку, який називається загальною шиною (рис. 11.2).

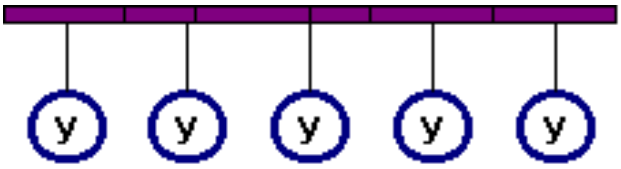



Рисунок 11.2 Топологія Загальна шина

Загальна шина дуже поширеною топологією для локальних мереж. Інформація, що передається, може поширюватися в обидві сторони. Застосування загальної шини знижує вартість проводки та уніфікує підключення різних модулів. Основними перевагами такої схеми є дешевизна та простота розведення кабелю по приміщеннях. Найсерйозніший недолік

загальної шини полягає в її низькій надійності: будь-який дефект кабелю або якогось із численних роз'ємів повністю паралізує всю мережу. Іншим недоліком загальної шини є її невисока продуктивність, тому що при такому способі підключення в кожний момент часу лише один комп'ютер може передавати дані до мережі. Тому пропускна спроможність каналу зв'язку завжди ділиться між усіма вузлами мережі.

2. **Топологія "Зірка"**. У цьому випадку кожен комп'ютер підключається окремим кабелем до загального пристрою, який називається концентратором, який знаходиться в центрі мережі (рис. 11.3).

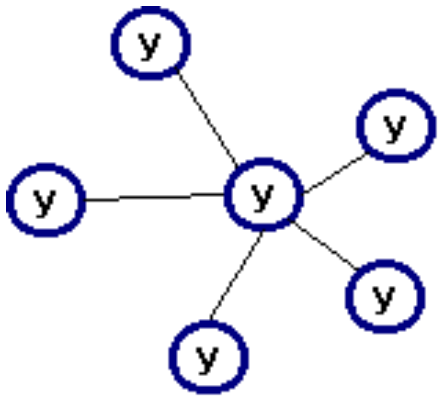



Рисунок 11.3 Топологія Зірка

У функції концентратора входить напрямок переданої комп'ютером інформації одному чи всім іншим комп'ютерам мережі. Головна перевага цієї топології перед загальною шиною - значно більша надійність. Будь-які неприємності з кабелем стосуються лише комп'ютера, до якого цей кабель приєднаний, і лише несправність концентратора може вивести з ладу всю мережу. Крім того, концентратор може відігравати роль інтелектуального фільтра інформації, що надходить від вузлів у мережу, та при необхідності блокувати заборонені адміністратором передачі. До недоліків топології типу зірка належить вища вартість мережного устаткування через необхідність придбання концентратора. Крім того, можливості нарощування кількості вузлів у мережі обмежуються кількістю портів концентратора. Нині ієрархічна зірка є найпоширенішим типом топології зв'язків як і локальних, і глобальних мережах.

3. **Топологія "Кільце"**. У мережах з кільцевою топологією дані в мережі передаються послідовно від однієї станції до іншої по кільцю, як правило, в одному напрямку (рис. 11.4). Якщо комп'ютер розпізнає дані як призначені йому, він копіює їх у внутрішній буфер. У мережі з кільцевою топологією необхідно вживати спеціальних заходів, щоб у разі виходу з ладу або відключення будь-якої станції не перервався канал зв'язку між іншими станціями. Перевага даної топології - простота управління, недолік - можливість відмови всієї мережі при збої у каналі між двома вузлами.

Рисунок 11.4 Топологія Кільце

4. **Комірчаста топологія.** Для комірчастої топології характерна схема з'єднання комп'ютерів, при якій фізичні лінії зв'язку встановлені з усіма комп'ютерами, що стоять поруч (рис. 11.5).

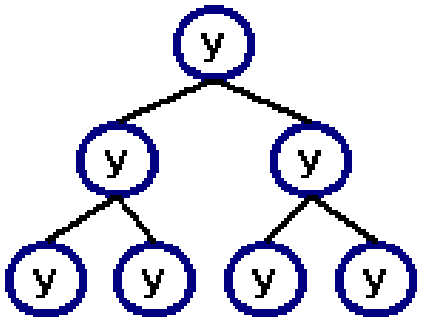


Рисунок 11.5 Комірчаста топологія

У мережі з пористою топологією безпосередньо зв'язуються ті комп'ютери, між якими відбувається інтенсивний обмін даними, а обміну даними між комп'ютерами, не з'єднаними прямими зв'язками, використовуються транзитні передачі через проміжні вузли. Комірчаста топологія допускає з'єднання великої кількості комп'ютерів і характерна, як правило, для глобальних мереж. Переваги цієї топології у її стійкості до відмов і перевантажень, т.к. є кілька способів обійти окремі вузли.

**5. Змішана топологія**. Тоді як невеликі мережі, зазвичай, мають типову топологію - зірка, кільце чи загальна шина, великих мереж характерна наявність довільних зв'язків між комп'ютерами. У таких мережах можна виділити окремі довільні підмережі, що мають типову топологію, тому їх називають мережами зі змішаною топологією (рис. 11.6).

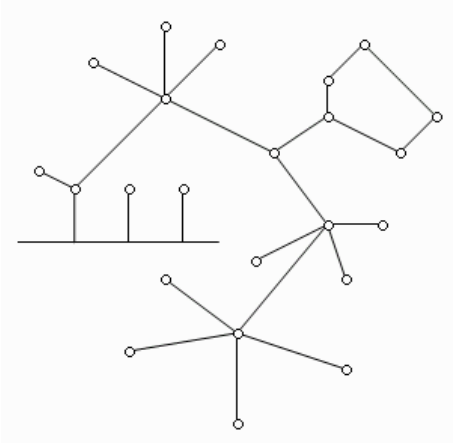


Рисунок 11.6 Змішана топологія

**Модель взаємозв'язку відкритих систем**

Основним завданням, яке вирішується під час створення комп'ютерних мереж, є забезпечення сумісності обладнання за електричними та механічними характеристиками та забезпечення сумісності інформаційного забезпечення (програм та даних) за системою кодування та форматом даних. Вирішення цієї задачі відноситься до галузі стандартизації і засноване на так званій моделі OSI (модель взаємодії відкритих систем - Model of Open System Interconnections). Модель OSI була створена на основі технічних пропозицій Міжнародного інституту стандартів ISO (International Standards Organization).

Відповідно до моделі OSI архітектуру комп'ютерних мереж слід розглядати різних рівнях (загальна кількість рівнів - до семи). Найвищий рівень – прикладний. На цьому рівні користувач взаємодіє з обчислювальною системою. Найнижчий рівень - фізичний. Він забезпечує обмін сигналами між пристроями. Обмін даними в системах зв'язку відбувається шляхом їхнього переміщення з верхнього рівня на нижній, потім транспортування і, нарешті, зворотним відтворенням на комп'ютері клієнта в результаті переміщення з нижнього рівня на верхній (рис. 11.7).

Для забезпечення необхідної сумісності кожному з семи можливих рівнів архітектури комп'ютерної мережі діють спеціальні стандарти, звані протоколами. Вони визначають характер апаратної взаємодії компонентів мережі (апаратні протоколи) та характер взаємодії програм та даних (програмні протоколи).

Фізично функції підтримки протоколів виконують апаратні пристрої (інтерфейси) та програмні засоби (програми підтримки протоколів). Програми, які підтримують протоколи, також називають **протоколами.**

Кожен рівень архітектури поділяється на дві частини:

- специфікацію послуг;
- специфікацію протоколу.


Рисунок 11.7 Модель взаємозв'язку відкритих систем

Специфікація послуг визначає, що робить рівень, а специфікація протоколу - як і це робить, причому кожен конкретний рівень може мати більше протоколу.

Розглянемо функції, що виконуються кожним рівнем програмного забезпечення:

1. **Фізичний рівень** здійснює з'єднання з фізичним каналом, так, від'єднання від каналу, керування каналом. Визначається швидкість передачі даних та топологія мережі.

2. **Канальний рівень** додає в масиви інформації, що передаються, допоміжні символи і контролює правильність переданих даних. Тут інформація, що передається, розбивається на кілька пакетів або кадрів. Кожен пакет містить адреси джерела та місця призначення та засоби виявлення помилок.

3. **Мережевий рівень** визначає маршрут передачі між мережами, забезпечує обробку помилок, а як і управління потоками даних. Основне завдання мережного рівня – маршрутизація даних (передача даних між мережами).

**4. Транспортний рівень** пов'язує нижні рівні (фізичний, канальний, мережевий) із верхніми рівнями, що реалізуються програмними засобами. Цей рівень поділяє засоби формування даних у мережі від засобів передачі. Тут здійснюється поділ інформації за певною довжиною та уточнюється адреса призначення.

**5. Сеансовий рівень** здійснює управління сеансами зв'язку між двома взаємодіючими користувачами, визначає початок та закінчення сеансу зв'язку, час, тривалість та режим сеансу зв'язку, точки синхронізації для проміжного

контролю та відновлення при передачі даних; відновлює з'єднання після помилок під час сеансу без втрати даних.

**6. Представницький** - управляє поданням даних у необхідній для програми користувача формі, виробляє компресію та декомпресію даних. Завданням цього рівня є перетворення даних під час передачі інформації у формат, який використовується в інформаційній системі. При прийомі даних цей рівень представлення даних виконує зворотне перетворення.

7. **Прикладний рівень** взаємодіє з прикладними мережевими програмами, що обслуговують файли, а також виконує обчислювальні, інформаційно-пошукові роботи, логічні перетворення інформації, передачу поштових повідомлень тощо. Головне завдання цього рівня – забезпечити зручний інтерфейс для користувача.

8. На різних рівнях обмін відбувається різними одиницями інформації: біти, кадри, пакети, сеансові повідомлення, повідомлення користувача.

**Мережеве обладнання**

Основними компонентами мережі є **робочі станції**, **сервери**, **середи передачі (кабелі)** та **мережеве обладнання**.

**Робочими станціями** називаються комп'ютери мережі, у яких користувачами мережі реалізуються прикладні завдання.

**Сервери мережі** - це апаратно-програмні системи, що виконують функції керування розподілом мережевих ресурсів загального доступу. Сервером може бути будь-який підключений до мережі комп'ютер, на якому знаходяться ресурси, що використовуються іншими пристроями локальної мережі. Як апаратна частина сервера використовується досить потужні комп'ютери.

Мережі можна створювати з будь-яким типом кабелю.

1. **Віта пара** (TP – Twisted Pair) – це кабель, виконаний у вигляді скрученої пари проводів. Він може бути екранованим та неекранованим. Екранований кабель стійкіший до електромагнітних перешкод. Віта пара найкраще підходить для малих установ. Недоліками даного кабелю є високий коефіцієнт загасання сигналу і висока чутливість до електромагнітних перешкод, тому максимальна відстань між активними пристроями ЛВС при використанні кручений пари повинна бути не більше 100 метрів.

2. **Коаксіальний кабель** складається з одного цільного або крученого центрального провідника, який оточений шаром діелектрика. Провідний шар алюмінієвої фольги, металевого обплетення або їх комбінації оточує діелектрик і служить одночасно як екран проти наведень. Загальний ізолюючий шар утворює зовнішню оболонку кабелю.

Коаксіальний кабель може використовуватися у двох різних системах передачі даних: без модуляції сигналу та з модуляцією. У першому випадку цифровий сигнал використовується в такому вигляді, в якому він надходить з ПК і відразу ж передається кабелем на приймальну станцію. Він має один канал

передачі зі швидкістю до 10 Мбіт/сек і максимальний радіус дії 4000 м. У другому випадку цифровий сигнал перетворюють на аналоговий і направляють його на приймальну станцію, де він знову перетворюється на цифровий. Операція перетворення сигналу виконується модемом; кожна станція має мати свій модем. Цей спосіб передачі є багатоканальним (забезпечує передачу по десятках каналів, використовуючи для цього лише один кабель). У такий спосіб можна передавати звуки, відеосигнали та інші дані. Довжина кабелю може досягати 50 км.

3. **Оптоволоконний кабель** є новішою технологією, що використовується в мережах. Носієм інформації є світловий промінь, який модулюється мережею і набуває форми сигналу. Така система стійка до зовнішніх електричних перешкод і, таким чином, можлива дуже швидка, секретна і безпомилкова передача даних зі швидкістю до 2 Гбіт/с. Кількість каналів у таких кабелях величезна. Передача даних виконується тільки в симплексному режимі, тому для організації обміну даними пристрою необхідно з'єднувати двома оптичними волокнами (на практиці оптоволоконний кабель завжди має парне, парне кількість волокон). До недоліків оптоволоконного кабелю можна віднести велику вартість, а також складність приєднання.

4. **Радіохвилі** в мікрохвильовому діапазоні використовуються як передавальне середовище в бездротових локальних мережах, або між мостами або шлюзами для зв'язку між локальними мережами. У першому випадку максимальна відстань між станціями становить 200 – 300 м, у другому – це відстань прямої видимості. Швидкість передачі - до 2 Мбіт/с.

**Бездротові локальні мережі** вважаються перспективним напрямом розвитку ЛЗ. Їх перевага - простота та мобільність. Також зникають проблеми, пов'язані з прокладанням та монтажем кабельних з'єднань – достатньо встановити інтерфейсні плати на робочі станції, і мережа готова до роботи.

Виділяють такі види мережного устаткування:

1. **Мережеві карти** – це контролери, що підключаються до слотів розширення материнської плати комп'ютера, призначені передачі сигналів у мережу і прийому сигналів з мережі.

2. **Термінатори** – це резистори номіналом 50 Ом, які виробляють загасання сигналу на кінцях сегмента мережі.

3. **Концентратори** (Hub) – це центральні пристрої кабельної системи чи мережі фізичної топології " зірка " , які за отриманні пакета однією зі своїх портів пересилає їх у всі інші. В результаті виходить мережа з логічною структурою загальної шини. Розрізняють концентратори активні та пасивні. Активні концентратори посилюють отримані сигнали та передають їх. Пасивні концентратори пропускають крізь себе сигнал, не посилюючи і відновлюючи його.

4. **Повторювачі** (Repeater) - пристрої мережі, що посилює і заново формує форму вхідного аналогового сигналу мережі на відстань іншого сегмента. Повторювач діє електрично для з'єднання двох сегментів. Повторювачі нічого не розпізнають мережеві адреси і тому не можуть використовуватися для зменшення трафіку.

5. **Комутатори** (Switch) - керовані програмним забезпеченням центральні пристрої кабельної системи, що скорочують мережевий трафік за рахунок того, що пакет аналізується для з'ясування адреси його одержувача і відповідно передається тільки йому. Використання комутаторів є дорожчим, а й більш продуктивним рішенням. Комутатор зазвичай значно складніший пристрій і може обслуговувати одночасно кілька запитів. Якщо з якоїсь причини потрібний порт зараз зайнятий, то пакет поміщається в буферну пам'ять комутатора, де і чекає своєї черги. Побудовані за допомогою комутаторів мережі можуть охоплювати кілька сотень машин і мати довжину кілька кілометрів.

6. **Маршрутизатори** (Router) - стандартні пристрої мережі, що працюють на мережному рівні і дозволяють переадресовувати та маршрутизувати пакети з однієї мережі в іншу, а також фільтрувати широкомовні повідомлення.

7. **Мости** (Bridge)- пристрої мережі, яке з'єднують два окремі сегменти, обмежених своєю фізичною довжиною, і передають трафік між ними. Мости також посилюють та конвертують сигнали для кабелю іншого типу. Це дозволяє розширити максимальний розмір мережі, одночасно не порушуючи обмежень на максимальну довжину кабелю, кількість підключених пристроїв або повторювачів на мережевий сегмент.

8. **Шлюзи** (Gateway) – програмно-апаратні комплекси, що з'єднують різнорідні мережі чи мережеві пристрої. Шлюзи дозволяє вирішувати проблеми розходження протоколів чи систем адресації. Вони діє на сеансовому, представницькому та прикладному рівнях моделі OSI.

9. **Міжмережеві екрани** (firewall, брандмауери) - це мережні пристрої, що реалізують контроль за інформацією, що надходить у локальну мережу і виходить з неї і забезпечують захист локальної мережі за допомогою фільтрації інформації. Більшість міжмережевих екранів побудовано на класичних моделях розмежування доступу, згідно з якими суб'єкту (користувачу, програмі, процесу або мережному пакету) дозволяється або забороняється доступ до будь-якого об'єкта (файлу або вузла мережі) при пред'явленні деякого унікального, властивого тільки цьому суб'єкту елемента.

# 12 ГЕОГРАФІЧНІ ІНФОРМАЦІЙНІ СИСТЕМИ

**ГІС** – сучасні комп'ютерні технології для картографування й аналізу об'єктів реального світу, подій і явищ, що відбуваються та будуть відбуватись у прогнозованому періоді. ГІС – це інформаційна система, яка забезпечує збір, збереження, обробку, доступ, відображення та поширення геопросторових даних.

**Геопросторові дані** – це дані, які ідентифікують географічне місце розташування та властивості природних або штучно створених об'єктів, а також їх межі на Землі. Ця інформація може отримуватися за допомогою GPS, дистанційного зондування Землі (ДЗЗ), картографування й різноманітних видів знімань тощо. Як свідчать спеціальні дослідження, 75–90 % усієї інформації, яку використовують спеціалісти різного рівня, містять у собі географічні (метричні, просторові) дані, тобто різні відомості про розподіл у просторі або по територіях об'єктів, явищ, процесів, подій (рис. 12.1). А робота з геоданими і є суттю ГІС.

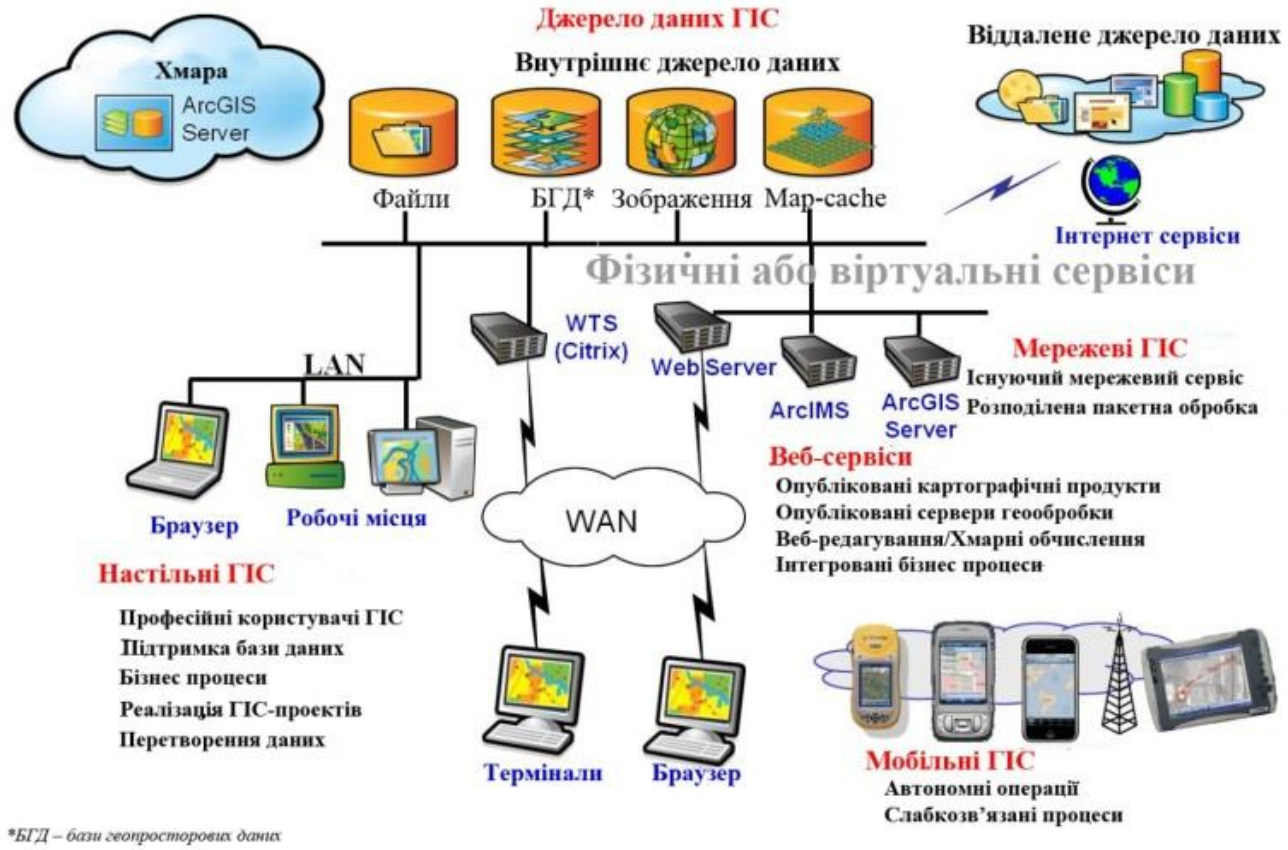



Рисунок 12.1 Географічна інформаційна система

**Предметом** ГІС є дослідження закономірностей інформаційного забезпечення користувачів, включаючи принципи побудови системи збору, накопичення, обробки, моделювання й аналізу просторових даних, їх відображення та використання, доведення до користувачів, формування технічних програмних засобів, розробки технології виготовлення електронних і цифрових карт, формування відповідних організаційних структур.

**Основні галузі застосування ГІС**. ГІТ розвивалися та розвиваються так само швидко, як і інші інформаційні технології з обробки даних. Фактично вони йшли тим же шляхом розвитку, що й будь-які інші інформаційні технології – введення даних, редагування даних, організація потоків даних, вихід в Інтернет.

Сьогодні ГІС працюють і на персональних комп'ютерах, і на серверах, і в Інтернет, і на КПК, і навіть на мобільних телефонах. Застосування ГІС можна знайти практично скрізь. Але найбільшу віддачу можна отримати там, де основне значення має просторовий характер інформації: в геології, екології, картографії, землекористуванні, військовій справі, – тут ГІС працюють уже давно і дуже успішно. Саме там створювалися перші ГІС.


Рисунок 12.2 Основні функції ГІС

У переліку геоінформаційних додатків можна виділити базові типи поширених завдань:

1) ***завдання обліково-інвентаризаційного типу***, де акцент робиться на даних, вимірах та оцінці "ступеня подібності" (задачі земельного кадастру,

підрахунки запасів природних ресурсів, управління розподіленою виробничою інфраструктурою). Це найпоширеніший тип додатків ГІС. Для додатків такого типу характерна робота з великою кількістю географічних об’єктів і висока детальність вивчення територій;

2) ***завдання планування розвитку, вибору маршрутів і управління перевезеннями***. Специфіка додатків цього типу пов’язана з нетрадиційними постановками оптимізаційних завдань, заданими на структурах припустимих шляхів;

3) ***моделювання і складні методи аналізу даних.*** Типове завдання – прогнозування повеней через аналіз водозбору та водостоку на заданому рельєфі.

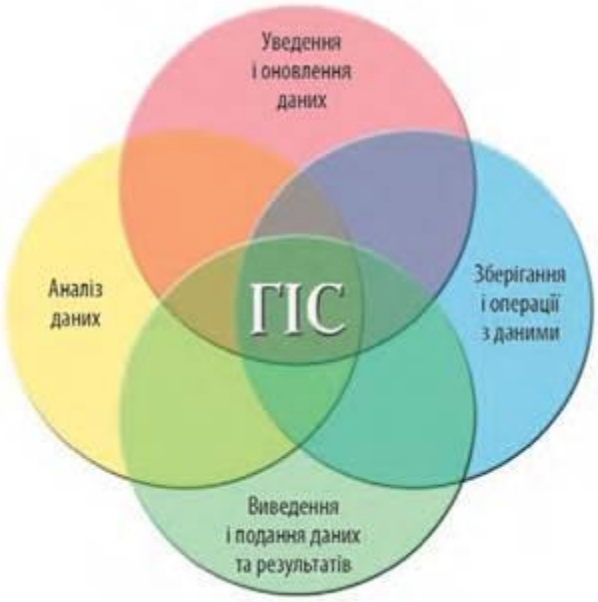



Рисунок 12.3 Переробка даних в ГІС

Як приклад розглянемо галузі, де застосування ГІС стало вже традиційним.

***Управління та планування розвитку територій***. Ця галузь ґрунтується на передбачуваній поведінці різних соціальних груп, які визначають суспільні потреби та можливості, що мають задане або передбачуване розміщення і динаміку в межах певної території.

***Містобудування й архітектура***. Проектування, інженерні вишукування, планування в містобудуванні, архітектурі. Це типова робота міських служб, що забезпечують нормальний розвиток підвідомчої території.

***Інженерна інфраструктура***. Інвентаризація, облік, планування розміщення об’єктів розподіленої виробничої інфраструктури (водопостачання, водовідведення, теплопостачання, газопостачання, електропостачання) та

управління ними, оцінка стану та прийняття рішень при ремонтних або аварійних ситуаціях.

***Управління земельними ресурсами, земельні кадастри***. Галузь характерна своєю власне географічною орієнтацією. Типовими задачами виступають складання кадастрів, класифікаційних карт, визначення меж ділянок, площ тощо.

***Управління природними ресурсами та природоохоронна діяльність***. Тут типовими проблемами є визначення поточних станів і запасів спостережуваних ресурсів, моделювання процесів у природному середовищі та побудова обґрунтувань рішень, прийнятих за управління засобами, що змінюють природні ресурси (Рис. 12.4).

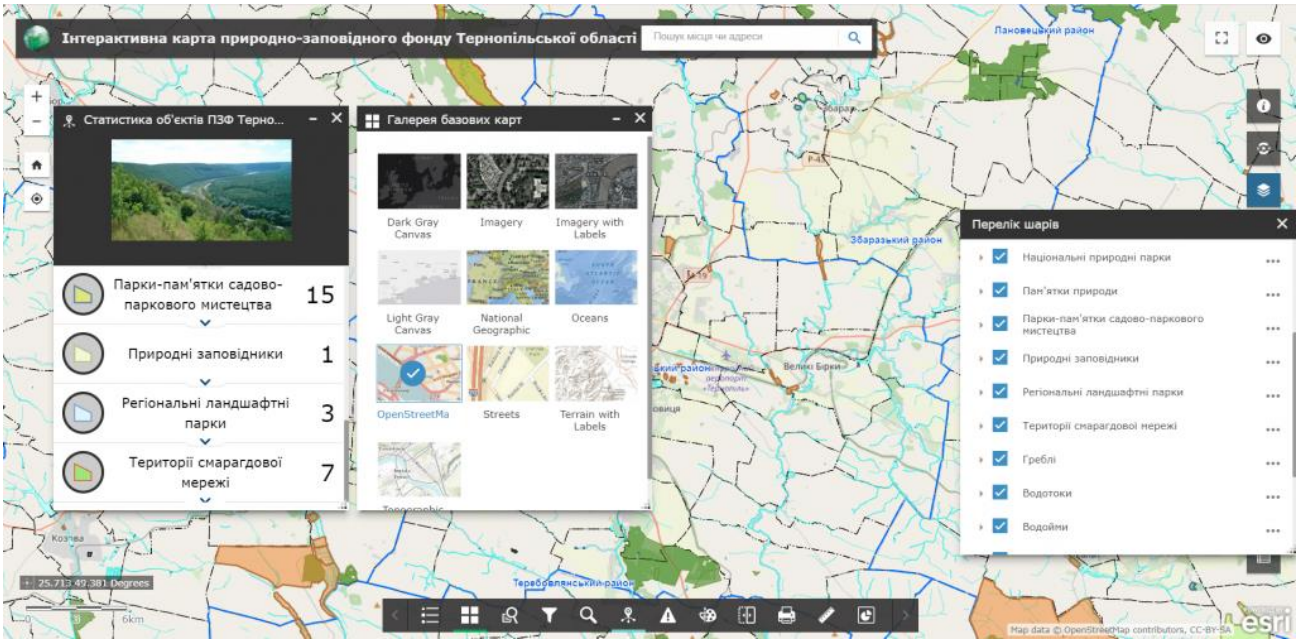


Рисунок 12.4 ГІС Управління природними ресурсами

***Геологія, мінерально-сировинні ресурси, горно-видобувна промисловість.*** Специфіка таких проблем полягає в тому, що потрібно розрахувати запаси корисних копалин на певній території за результатами визначення в окремих точках (розвідницьке буравлення, пробні шурфи тощо) при відомій моделі процесу утворення родовища.

***Планування й управління перевезеннями (логістика).*** На карті задаються пункти збереження вантажів зі своїми характеристиками, пункти, що очікують потрібні їм вантажі; засоби перевезення зі своїми характеристиками, позицією, станом і спеціалізацією; мережа доріг зі своїми характеристиками (середня швидкість, ремонти, об’їзди, пробки, границі, митні пункти тощо).

Потрібно скласти план перевезень і корегувати його при виникненні непередбачених ситуацій.

***Наземне, аеро- та гідронавігаційне картографування й керування наземним, повітряним і водним транспортом.*** Традиційні галузі зі зрозумілими завданнями. Особливе місце займають завдання керування рухомими об’єктами за умовою виконання заданою системою відношень між ними та нерухомими об’єктами (рис. 12.5).

***Маркетинг і аналіз ринку***. Визначення тенденцій розвитку ситуації, оцінка впливу різних топологічних властивостей – близькості, перетинань і суміщень різних ареалів на їх взаємодію, урахування різних умов визначених на об’єктах із заданими позиціями, потребами та можливостями їх розвитку.

***Сільське господарство***. Підрахунок запасів ресурсів за низкою точкових вимірів, планування перевезень, взаємодія динаміки зміни ареалів, категоризація та виділення "подібності" просторових об’єктів, точне землеробство.

***Надзвичайні ситуації***. Облік потенційно небезпечних об’єктів, моделювання наслідків у надзвичайних ситуаціях.

***Служби швидкого реагування.*** Суспільна безпека, пожежогасіння, швидка медична допомога.


Рисунок 12.5 Моніторинг, моделювання та прогнозування стану довкілля

Географічні об’єкти в ГІС подаються як набір просторових і атрибутивних даних із загальною назвою географічні дані (Geographic data).

Географічні дані – це дані, які описують певну частину поверхні Землі або об’єкти, що знаходяться на цій поверхні. Вони показують об’єкти з точки зору

розташування їх на поверхні Землі, тобто представляють собою географічну карту місцевості.

Просторові дані – це дані про місце розташування об’єктів або поширенню явищ, які представлені в певній системі координат у словесному або числовому опису.

Географічні дані (геодані) містять чотири інтегрованих компоненти:

– географічне положення (розміщення) просторових об’єктів подається 2-х, 3-х або 4-мірними координатами в географічно співвіднесеній системі координат (широта / довгота);

– атрибути (семантичні дані) − властивість, якісна або кількісна ознака, яка характеризує просторовий об’єкт, але не пов’язана з його місцем розташування;

– просторові відношення − внутрішні взаємовідношення між просторовими об’єктами (наприклад, напрямок об’єкта А щодо об’єкта В, відстань між об’єктами А і В, вкладеність об’єкта А в об’єкт В тощо);

– часові характеристики подаються в вигляді строків одержання даних, визначають їх життєвий цикл (строків придатності або достовірності), зміну місця розташування або властивостей просторових об’єктів у часі.

Геодані – це дані про локальні просторові властивості: місце розташування, форму, розміри та просторові відношення географічних об’єктів, явищ, процесів у реальному земному просторі.

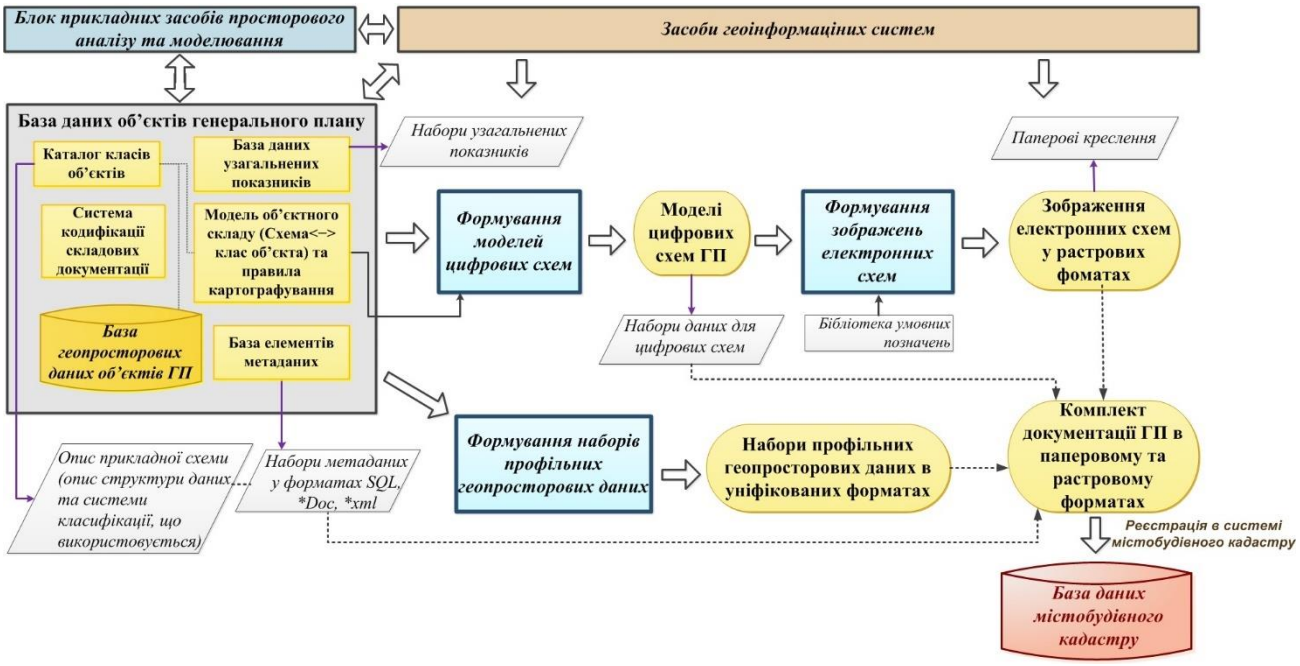



Рисунок 12.6 Перетворення інформації в ГІС системах

До них відносяться:

• географічна широта та довгота;

• прямокутні координати;

• поштові адреси; поштові індекси й інші коди, які ідентифікують попередньо поділені ділянки території;

• місцеположення, зафіксоване на карті або в просторі.

За змістовністю геодані поділяються на базисні (координатні) та спеціальні (тематичні, атрибутивні). Просторові характеристики визначають положення об’єкта в заздалегідь визначеній системі координат. Координатна компонента характеризує місце розташування, форму, розміри, просторові відношення об’єктів дослідження в ГІС.

Атрибутивні (семантичні) дані – це дані, якими описується змістовна, значеннєва інформація про географічні об’єкти, властивості географічних об’єктів.

Властивість – це категорія, яка обумовлює відмінність або спільність об’єкта з іншими об’єктами і виявляє себе при зіставленні різних об’єктів.

Географічна інформаційна система – система обладнання, програмного забезпечення, даних, людей, організацій та інститутських домовленостей для збору, збереження, аналізу й поширення інформації про території Землі.

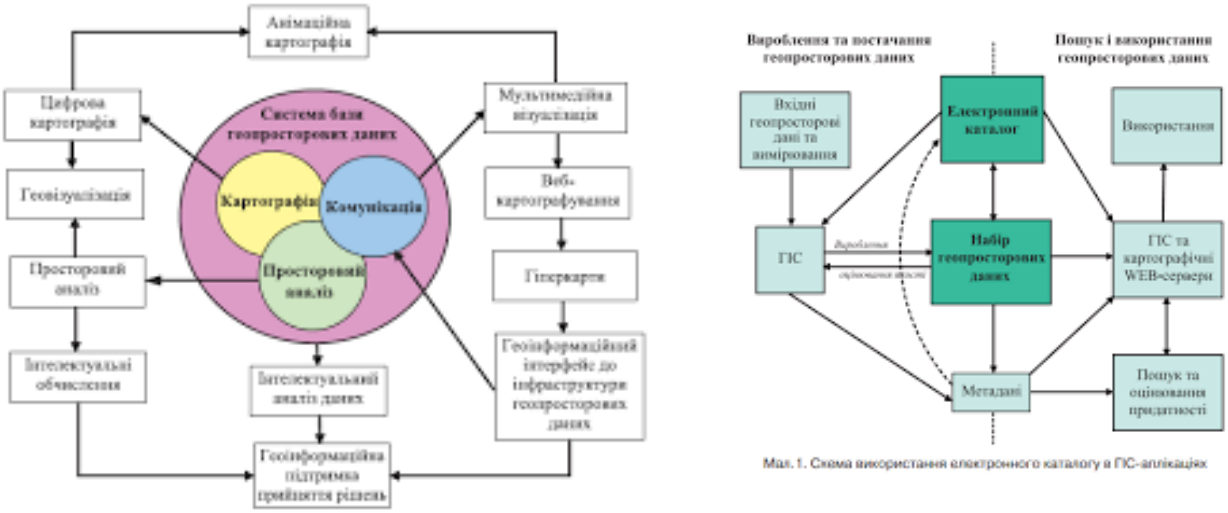



Рисунок 12.7 Архітектура ГІС

Основними складовими геоінформатики є:

1. Загальна геоінформатика – розділ геоінформатики, що вивчає загальні властивості просторової інформації без конкретизації її змісту, займається дослідженням і розробкою наукових засад, концепцій, узагальненим аналізом безвідносно до її прикладного характеру.

2. Прикладна геоінформатика – розділ геоінформатики, що вивчає практичні методи робіт з ГІС і ГІТ.

3. Спеціальна геоінформатика – розділ геоінформатики, що слугує основою для аналізу систем і методів обробки просторових даних.

Геоінформаційні технології (ГІТ) – сукупність методів і прийомів практичного використання досягнень геоінформатики для маніпулювання просторовими даними, їх подання й аналізу.

Компоненти ГІС – частини системи, виділені за певною ознакою або сукупністю ознак, що розглядаються як єдине цілісне утворення.

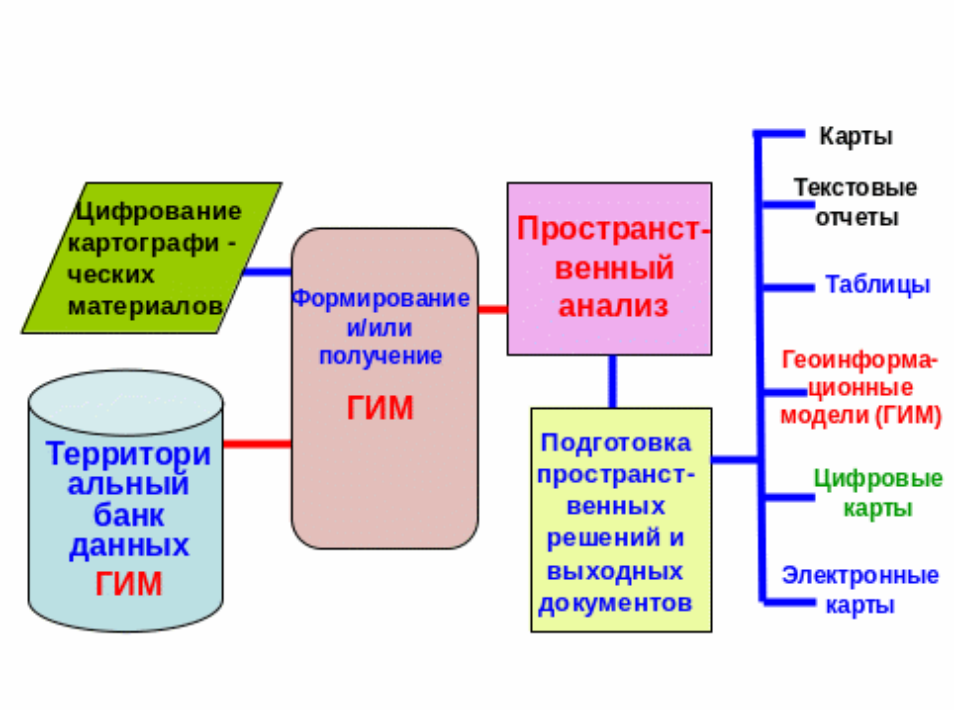



Рисунок 12.8 Забезпечення ГІС

Інформаційне забезпечення ГІС є відповідним чином закодованою просторовою інформацією, тобто інформацією, пов’язаною з місцем її розташування. Просторові дані складаються з цифрових представлень реально існуючих дискретних просторових об’єктів (процесів, явищ, подій).

Апаратне забезпечення – це комп’ютер, на якому інстальована ГІС, а також набір периферійних пристроїв, що забезпечують уведення і виведення інформації. ГІС можуть працювати на різних типах апаратних комп’ютерних платформ від централізованих серверів до окремих персональних або пов’язаних мережею настільних комп’ютерів.

Програмне забезпечення – це сукупність взаємопов’язаних програмних модулів, які забезпечують виконання основних функцій ГІС (можливість

уведення, збереження, обробки та аналізу даних, їх візуалізації, надання підтримки прийняття рішень), а також безпосереднє керування ГІС у цілому.

Кадрове забезпечення ГІС складають як кваліфіковані технічні фахівці, які розробляють і підтримують системи, створюють і керують даними, так і безпосередньо користувачі, які використовують їх у повсякденній роботі. Від кадрового забезпечення залежить ефективність ГІС.

Функціональне забезпечення – методологічний апарат, закладений у ГІС. Сучасні ГІС включають засоби розробки, які дозволяють нарощувати функціональність і перетворювати універсальні ГІС у спеціалізовані системи для конкретних галузей, сфер знання, робочих колективів. Одним з головних принципів побудови карт є шарова структура зображень, яка дозволяє комплексно представляти вихідні зображення (рис. 12.9).


Рисунок 12.9 Представлення зображень у вигляді шарів

Характерною особливістю ГІС є наявність картографічної основи – цифрових карт, планів і схем, графічні об’єкти яких пов’язані посиланнями з елементами даних зовнішніх інформаційних джерел (просторові дані інших ГІС, атрибутивні дані серверів баз даних, бази знань, тексти і гіпертексти, звукові та відеофайли, анімація, зображення і будь-які структуровані документи.

ГІС інтегрують не тільки дані але й передові інформаційні технології: технології автоматизованої картографії; баз даних; цифрової обробки результатів позиціонування та дистанційного зондування Землі (ДЗЗ); геоінформаційних web-сервісів у відкритих середовищах Інтернет та інших технологій, які не тільки розширюють можливості ГІС, але й визначають майбутню стратегію розвитку ГІС у зв’язку з інтеграцією та розвитком передових інформаційних технологій.

Крім того, до відмінностей ГІС від інших ІС можна віднести те, що це:

– людино-програмно-машинний комплекс із прийому, обробки, збереження, аналізу та передачі будь-якої просторово-розподіленої інформації;

– можливість оперативного реагування на будь-яку ситуацію, яка виникла на будь-якій території, з отриманням за нею всієї необхідної картографічної і тематичної інформації;

– можливість накладання різноманітної тематичної інформації на той самий просторовий контур й отримання нової інформації про територію;

– аналітичне, картометричне дослідження й аналіз з одночасною побудовою будь-яких карт, планів і схем;

– моделювання тих чи інших процесів, явищ і вивчення змін стану в часі;

– візуалізація просторової інформації й можливість її подання в динамічному режимі (рис. 12.10);

– управління ресурсами та територіями;

– швидкість, якість і точність;

– поєднання науки, технології та бізнесу;

– новий світогляд і нове мислення, побудовані на просторовій ідеології.

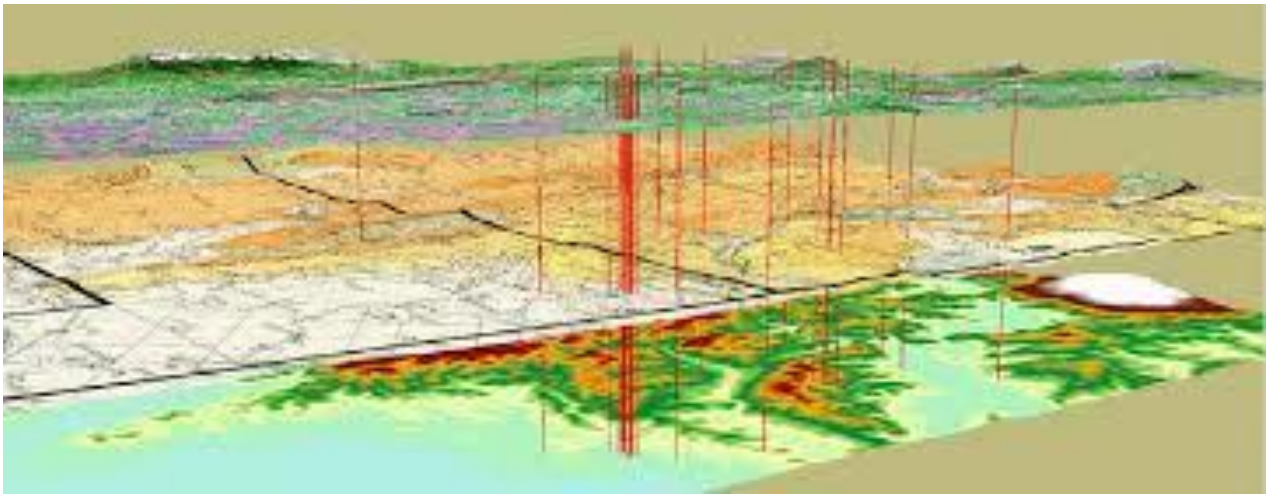


Рисунок 12.10 Представлення візуальної інформації

***Екологічний стан навколишнього середовища.*** Карти відбивають комплексну оцінку стану і рівня забруднення навколишнього середовища й окремих компонентів природи, систему моніторингу, природно-заповідний фонд та інші території, що охороняються.

Важливим моментом необхідно вважати посилення міжнародної кооперації та координації геоінформаційної діяльності. Одним із головних наслідків є створення глобальних інформаційних систем типу Глобальної природно-ресурсної бази GRID під егідою ЮНЕСКО, ГІС європейського економічного співтовариства CORINS і багато інших (рис. 12.11).

Рисунок 12. 11 Використання ГІС в екологічних дослідженнях

Про значення ГІС можна говорити, виходячи з того, яка увага приділяється їм у більшості країн світу. У багатьох із них створені національні й регіональні організації, до завдань яких входить розвиток досліджень, пов’язаних із ГІС, розробка пропозицій у сфері національного й міського планування інформації, координація програм отримання, обробки й розповсюдження цієї інформації, створення мереж ГІС. З цією метою розроблена правова база, створюється потужне, апаратне й програмне забезпечення, налагоджена підготовка та перекваліфікація необхідного класу спеціалістів.

Розвиток інформаційних технологій на базі обчислювальної техніки, створення автоматизованих, високопродуктивних робочих станцій, банків даних і баз знань, а також обчислювальних мереж привело до появи нового напрямку в інформатиці – геоінформатики, в основі якої лежать ГІС і ГІТ. На сьогодні вже існують (функціонують) сотні ГІС різного рівня.

Для раціоналізації управлінських процесів у органах виконавчої влади і місцевого самоврядування в розвинутих державах і країнах з перехідною економікою запроваджуються новітні інформаційні технології, в тому числі ГІТ, оскільки відомо, що вони можуть широко застосовуватися в практиці управління, відкривають нові можливості при аналізі територіально-прив’язаних даних, пошуку закономірностей їх розподілу, моделюванні та прогнозі ситуацій.

ГІС загальнодержавного і регіонального рівнів доцільно орієнтувати на вирішення завдань:

1) оцінки стану природних ресурсів регіону та планування процесів їх раціонального використання;

2) оцінки та прогнозування екологічного стану регіону;

3) оцінки ефективності і прогнозування розвитку регіонів;

4) розробки рекомендацій щодо забезпечення функцій управління регіонами.

Необхідно відзначити, що ці проблемні сфери є також домінуючими для регіональних прикладних дистанційних досліджень регіонів з космосу. Тому можна стверджувати про важливість ГІС і ГІТ для перспективних інформаційних технологій у прикладних дистанційних дослідженнях територіальних угруповань з космосу.

# 13 КАДАСТРОВІ ІНФОРМАЦІЙНІ СИСТЕМИ

Державними кадастрами природних ресурсів називається звід економічних, екологічних, організаційних і технічних показників, що характеризують якість і кількість природного ресурсу, склад і категорії користувачів. Дані кадастрів служать забезпечення раціонального використання природних ресурсів та охорони навколишнього середовища від шкідливих впливів (Рис. 13.1). На основі кадастрів проводиться грошова оцінка природного ресурсу, його продажна ціна, система заходів з відновлення порушеного стану природи. Важливо, щоб дані про якісні характеристики природних ресурсів, що містяться у відповідних кадастри, служили основою при прийнятті рішення про надання природного ресурсу у користування.


Рисунок 13.1 Значення кадастру

Сучасні кадастрові системи створюються багатоцільовими. На них, окрім традиційних юридичних і фіскальних функцій, покладається забезпечення інформаційної підтримки прийняття рішень у сфері охорони та раціонального використання природних ресурсів і сталого розвитку територій.

До основних принципів створення багатоцільових кадастрів належать: мультиспрямованість, мультизастосовність, мультиучасть, інтегрованість, розподіленість, масштабованість та інтероперабельність. Перша трійка принципів продиктована вимогами до розширеного вмісту кадастрових даних,

що задовольняє міжгалузеве їх використання, а також визначає основний спосіб ефективної підтримки їх в актуальному стані. Вони ґрунтуються на відомостях із першоджерел, підготовлених багатьма установами, які відповідають за створення і постачання тих чи інших інформаційних ресурсів, включених до складу кадастрових даних, наприклад: топографічні карти, реєстри фізичних та юридичних осіб, реєстри адрес, реєстри зон містобудівних та інших обмежень. Решту принципів можна віднести до групи технологічного забезпечення реалізації багатоцільового кадастру як розподіленої мережі інформаційних систем, в якій здійснюється інтегрування даних з різних джерел на основі використання однорідних за архітектурою, інформаційно й функціонально сумісних (інтероперабельних) систем.

Процес реформування кадастрів ґрунтується на широкому залученні геоінформаційних систем і технологій та цифрових моделей і методів збирання, накопичення та використання кадастрових даних. Практика переконує в необхідності їх побудови за архітектурою відкритих систем з чітко визначеними уніфікованими структурними компонентами, програмними сервісами зі стандартизованими інтерфейсами взаємодії та наборами вхідних і вихідних електронних документів (е-документів). За такими принципами створювалася та розвивається надскладна суперсистема сучасності, якою є глобальна інформаційна мережа Інтернет, а також інші глобальні та регіональні системи типу систем управління авіаперевезеннями вантажів і пасажирів, банківських систем (Рис. 13.2)

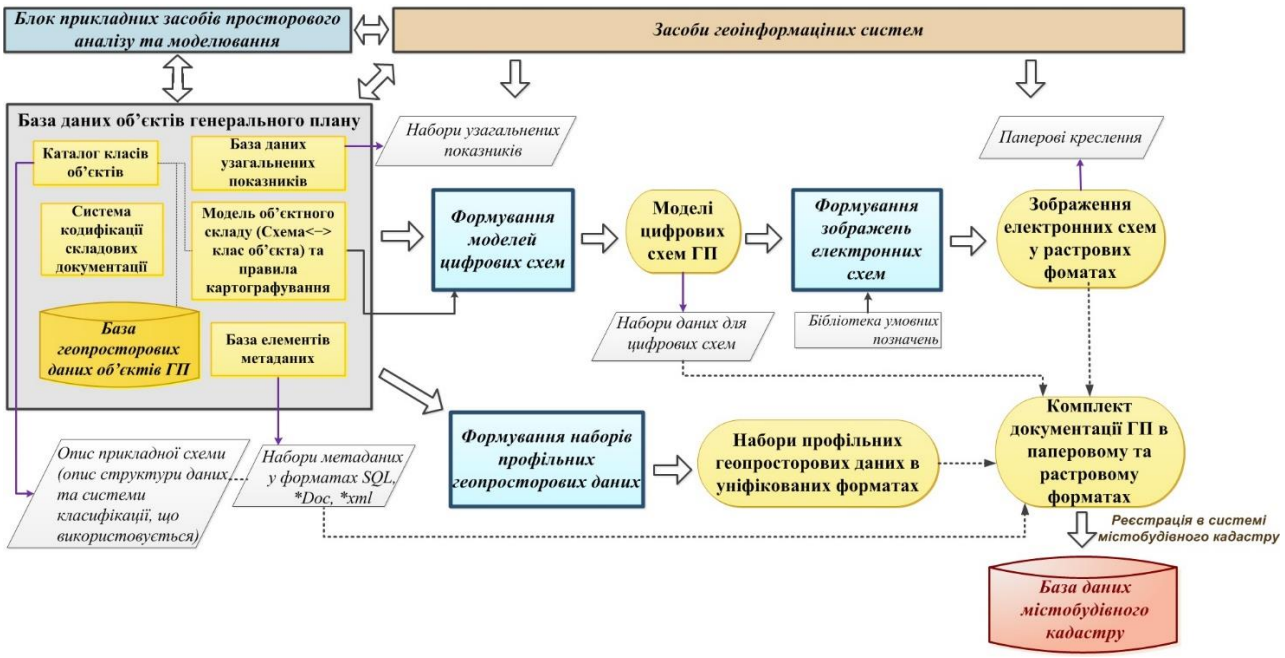



Рисунок 13.2 Технологія розроблення кадастру

Основу комплексної автоматизації процесів збирання, реєстрування, зберігання та використання даних у сучасних кадастрових системах, безперечно, складають наскрізні геоінформаційні технології (Рис. 13.3).

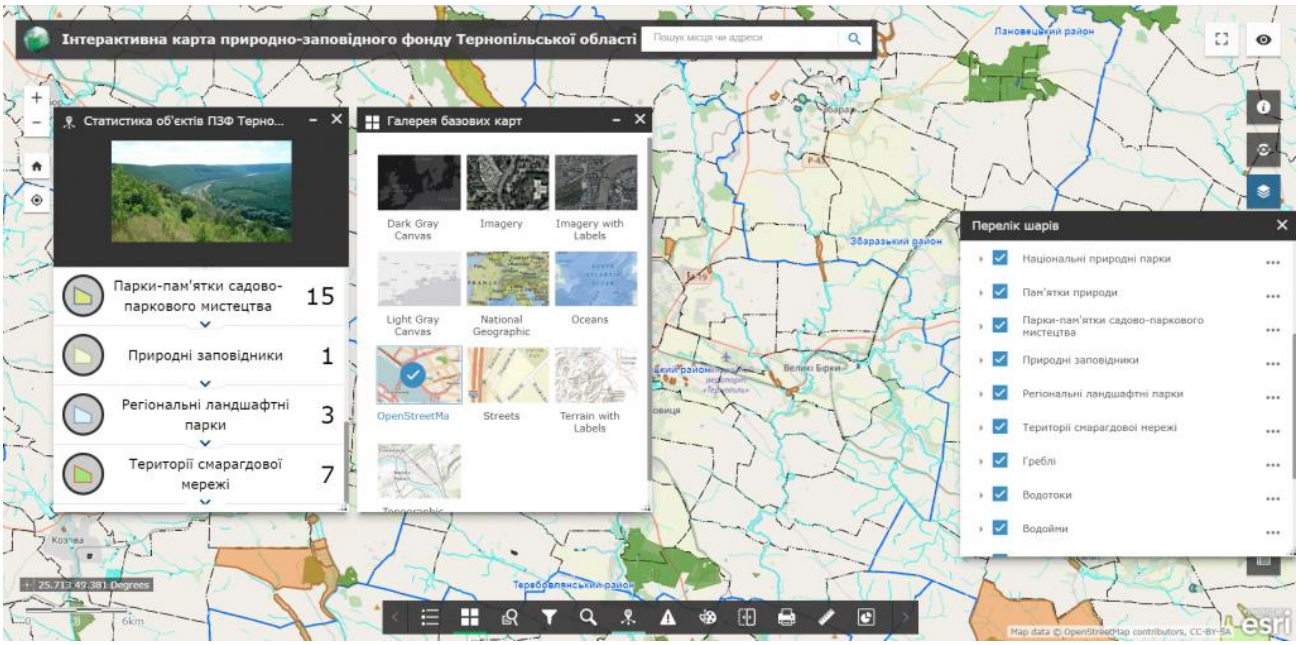



Рисунок 13.3 Геоінформаційна система управління природними ресурсами

Для автоматизації окремих процесів використовуються цифрові засоби отримання просторових даних, бази даних та програмні сервіси з уніфікованими функціями та інтерфейсами. За такого підходу створюються умови для інтероперабельності окремих програмних сервісів, які розробляються і постачаються різними виробниками, а також для здорової конкуренції у сфері розроблення програмних засобів, що в свою чергу сприяє підвищенню функціональності, надійності, постійному технічному удосконаленню програмних засобів та зниженню їх вартості.

Аналогічно вирішуються проблеми уніфікації міжсистемних зв’язків різних кадастрів в інфраструктурі геопросторових даних (наприклад, земельного і містобудівного, природних ресурсів і земельного) та передачі кадастрових даних в інші інформаційні системи (наприклад, інформаційні системи ринку нерухомості або інформаційні системи земельного банку тощо) (Рис. 13.4).

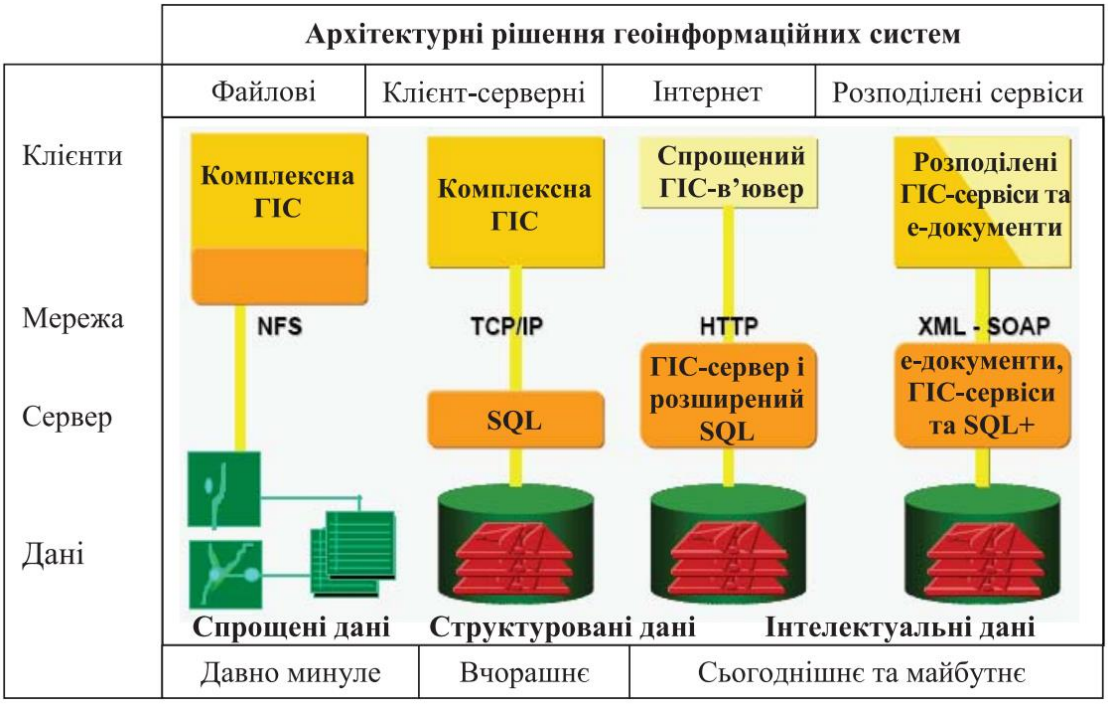



Рисунок 13.4 Сервіс-орієнтована архітектура кадастрових систем. NFS (Network File System) – мережна файлова система; TCP/IP (TCP: Transmission Control Protocol – Протокол керованого передавання; IP: Internet Protocol – міжмережний протокол) – маршрутизований мережний протокол; HTTP (Hyper Text Transfer Protocol) – протокол передавання гіпертекстових документів; XML (eXtensible Markup Language) – розширювана мова розмітки; SOAP (Simple Object Access Protocol) – простий протокол доступу до об'єктів. SOAP розроблено для реалізації віддаленого виклику процедур у розподілених системах. Наразі він розглядається як загальний протокол обміну структурованими повідомленнями у форматі XML в розподіленому обчислювальному середовищі, а не тільки для виклику процедур

Пропоновані сервіс-орієнтована архітектура кадастрової ГІС та система уніфікованого електронного кадастрового документообігу призначені для реалізації наскрізної інформаційної технології реєстрації та оброблення даних про земельні ділянки, права та їх обмеження від реєстрації вхідних документів (заяв громадян та юридичних осіб, технічних звітів з даними про земельні ділянки в обмінних форматах, додаткових документів, що посвідчують права, особу та інше) до формування й ведення бази даних земельного кадастру на всіх територіально розподілених рівнях АІС ДЗК (базовому на територію міст і районів, регіональному (обласному) та загальнодержавному). Така технологія забезпечує оброблення й інтегрування даних з різних джерел, які вводяться в систему на початкових етапах реєстрації об'єкта нерухомості, а після контролю й під твердження передаються в базу даних реєстрів земельного кадастру для довгострокового зберігання й використання в системах управління земельними ресурсами та обслуговування земельного ринку. Сервіс-орієнтована архітектура розподіленої АІС ДЗК (Рис. 13.5) має трирівневу логічну структуру в складі:

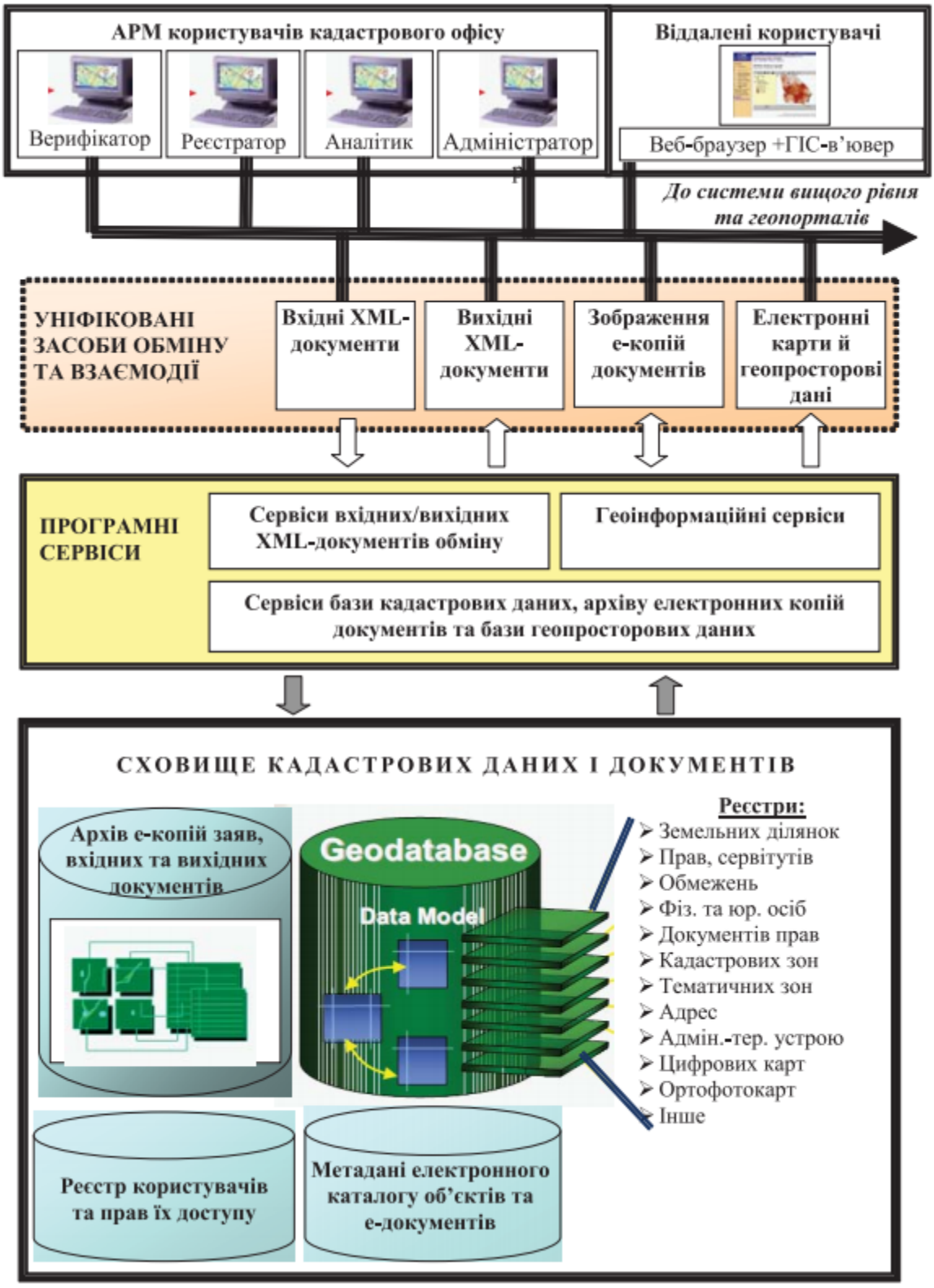



Рисунок 13.5 Сервіс-орієнтована архітектура автоматизованої інформаційної системи державного земельного кадастру

сервера бази даних для сховища кадастрових да них і документації, геопросторових даних, цифрових карт, планів та ортофотозображень;

програмних сервісів, у т. ч. для: обслуговування архіву електронних копій заяв та вхідних даних; створення і ведення бази даних реєстрів земельного кадастру; оброблення геопросторових даних і формування електронних карт; формування та інтерпретації уніфікованих е-документів обміну кадастровими даними;

спеціального програмного забезпечення клієнтських автоматизованих робочих місць (СПЗ АРМ), зокрема: приймання заяв та інших вхідних документів про реєстрацію земельних ділянок, прав і обмежень; адміністрування системи та інформаційного обслуговування і взаємодії з АІС ДЗК вищого рівня, геопорталами та іншими інформаційними системами органів державної влади й органів місцевого самоврядування та інших зацікавлених організацій.

Ключовою компонентою у пропонованій архітектурі АІС ДЗК є сховище кадастрових даних і документів у середовищі однієї із сучасних систем керування базами геопросторових даних (СКБГД), що реалізується на основі розширення традиційних об’єктно-реляційних СКБД засобами роботи з геопросторовими даними, наприклад: Oracle має роз ширення Oracle Spatial 10g, IBM DB2 – Spatial Extender, PostgreSQL – PostGis, MySQL версія 5.0.0 та пізніші, ЛИНТЕР версія 6.0.10 та інші.

У цих СКБГД уведено спеціальний абстрактний тип даних "Геометрія" для відображення просторових властивостей об’єктів, спеціальні механізми просторових індексів для ефективного доступу до геопросторових даних та розширена мова SQL для формування просторових запитів в операціях аналізу просторових і топологічних відношень між об’єктами. Тепер більшість задач геоінформаційного моделювання й аналізу може вирішуватися безпосередньо в середовищі СКБГД, а не тільки засобами інструментальних ГІС. СКБГД реально забезпечує такі важливі для кадастрових ГІС принципи процесу оброблення геопросторових даних: цілісність даних, спільний доступ до даних, адміністрування та розмежування доступу до даних, масштабованість, незалежність від ГІС-платформ, безпека даних, паралельність запитів, розподіленість баз даних, реплікація даних. Основна ж перевага застосування СКБГД у кадастрових ГІС полягає в забезпеченні цілісності кадастрових даних у сенсі зберігання та маніпулювання обома складовими моделей геопросторових об’єктів (геометричними та атрибутивними) в єдиному середовищі та незалежно від форматів інструментальних ГІС.

Обмін геопросторовими даними СКБГД із зовнішніми компонентами реалізується на основі географічної мови розмітки GML за стандартом ISO/DIS 19136: Geographic Information Geography Markup Language (GML), а для безпосереднього доступу внутрішніх програмних засобів до СКБГД

застосовується розширена мова SQL і два відкриті формати (WKB – well-known binary та WKT – well-known text), які рекомендовані у стандартах консорціуму OGS як засоби уніфікації доступу до баз геопросторових даних.

Застосування СКБГД для зберігання й оброблення геопросторових даних у кадастрових ГІС забезпечує незалежність прикладних програм та кадастрових даних (найдорожчих і найбільш трудомістких компонентів системи) від конкретних ГІС-платформ та можливість спільного використання цих даних при роботі в середовищі різних інструментальних ГІС, оскільки в нових версіях ГІС практично усіх провідних компаній (ArcGIS, AutoCADMap, MapInfo) та в ГІС з відкритими кодами (gvGIS, QGIS та ін.) реалізовані компоненти прямого доступу до СКБГД.

У сховищі кадастрових даних і документів (див. мал.2) виділено такі основні розділи:

архів електронних копій заяв землекористувачів, вхідних та вихідних документів, що відповідно подаються на реєстрацію в системі та формуються і видаються за допомогою системи;

власне база геопросторових даних кадастру (Geodatabase) з традиційними реєстрами (земельних ділянок та угідь; фізичних та юридичних осіб як суб'єктів прав на землю; прав, сервітутів, оренди та обмеження прав на земельні ділянки; документів, що посвідчують права; кадастрових зон, кадастрових кварталів, інших територіальних зон та меж об'єктів адміністративно-територіального устрою України; вулиць і адрес населених пунктів);

метадані електронного каталогу об'єктів бази кадастрових даних, їх атрибутів та шаблонів е-документів;

реєстр користувачів системи та прав їх доступу до програм і кадастрових даних.

Основу електронного документообігу кадастру в сучасних системах складають електронні скан-копії заяв та інших вхідних паперових документів, а також структуровані е-документи у форматах мови XML, серед яких можна виділити такі групи документів:

е-документи імпорту даних з обмінних файлів системи інвентаризації земельних ділянок, землевпорядної документації або інших джерел та взаємодіючих інформаційних систем (містобудівного, лісового, водного кадастрів, реєстрів фізичних та юридичних осіб тощо);

внутрішні е-документи, що формуються в процесі роботи користувачів із системою, включаючи XML-документи про реєстрацію користувачів на початку кожного сеансу роботи в системі та всі запити (звернення) і дії користувачів з кадастровими даними, а також е-документи відповідей системи на ці запити;

е-документи експорту кадастрових даних користувачам АІС ДЗК, в т. ч. і в інші взаємодіючі інформаційні системи та системи підтримки прийняття управлінських рішень.

Данна система земельного кадастру є великою інформаційною системою, яка розробляється по замовленню держави. Але є потреба і менш складних системах, які і вартують значно менше, і вирішують локальні проблеми. До такої відноситься інформаційна система кадастру природних лікувальних ресурсів, зокрема таких як рапа і пелоїд Куяльницького лиману. Така система розроблялась в ОДЕКУ на каф. Інформатики і деякі результати аспекти розробки наведені у даній лекції.

Актуальність задачі полягає в вирішенні задачі створення, запровадження та експлуатації програмно-інформаційних комплексів забезпечення автоматизованої системи ведення державного кадастру природних лікувальних ресурсів.

Метою дослідження є створення інформаційної системи ведення кадастру природно-лікувальних ресурсів, що мінімізує неоднозначність сховища даних для розподілених джерел і споживачів даних.

Передумовою досягнення цієї мети є те, що при використанні не локальної БД, а такої яка розташовується на сервері, а для обробки і відображення використовується web-сайт, причому за відсутності доступу до мережевих ресурсів, абоненти можуть накопичувати необхідні ресурси і функціонувати як локальні.

Дані Кадастру застосовуються для:

1) здійснення ефективного поточного і перспективного використання природних лікувальних ресурсів у санаторно-курортному лікуванні, медичній реабілітації, рекреації населення;

2) забезпечення раціонального видобутку, використання та охорони природних лікувальних ресурсів;

3) створення сприятливих умов для лікування, профілактики захворювань та відпочинку людей;

4) удосконалення системи проведення природоохоронних заходів, створення округів і зон санітарної (гірничо-санітарної) охорони курортів;

5) вирішення інших питань, пов'язаних з використанням природних лікувальних ресурсів.

Згідно з Постановою Кабінету Міністрів України від 26 липня 2001 р. № 872 в Кадастр включаються відомості у формі текстових, цифрових і графічних (картографічних) матеріалів за видами (типів) природних лікувальних ресурсів: мінеральні і термальні води, лікувальні грязі та озокерит, ропа лиманів та озер, морська вода, природні об'єкти і комплекси із сприятливими для лікування, медичної реабілітації та профілактики захворювань кліматичними умовами.

***Актуальність задачі.*** При всій важливості перерахованих задач в даний момент не вирішена задача створення, запровадження та експлуатації

програмно-інформаційних комплексів забезпечення автоматизованої системи ведення державного кадастру природних лікувальних ресурсів.

Проведений аналіз показав, що ця задача не вирішена, бо всі дані кадастру існують у паперовому варіанті у вигляді бібліотечного каталогу або в електронному вигляді, у вигляді документів MS Word, що не суттєво відрізняється від паперового аналога.

Даний варіант розв'язання задачі має багато недоліків:

1) дуже важко робити пошук даних;
2) важко робити аналіз даних;
3) схильність до багатьох помилок;
4) багато часу займає створення і заповнення нових даних;
5) неактуальність інформації;
6) неоднозначність інформації;
7) відсутність швидкого та відкритого доступу.

Існуючі електронні кадастри будуються за двома основними принципами: створення великих інформаційних систем кадастрів (наприклад, земельних), для яких характерні значні програмні та апаратні ресурси для їх реалізації і, відповідно, дуже висока вартість. Другим підходом є створення локальної інформаційної системи, розташованої на одному комп'ютері (наприклад, демоверсія Інституту курортології), яка ніяк не пов'язує дані, отримані від різних джерел, отже, практично повторює паперовий варіант сховища даних, хоча поширення і сприйняття даних ПЛР істотно поліпшується.

***Метою дослідження*** є створення інформаційної системи ведення кадастру природно-лікувальних ресурсів, що мінімізує неоднозначність сховища даних для розподілених джерел і споживачів даних.

Передумовою досягнення цієї мети є те, що при використанні не локальної БД, а такої яка розташовується на сервері, а для обробки і відображення використовується web-сайт, причому за відсутності доступу до мережевих ресурсів, абоненти можуть накопичувати необхідні ресурси і функціонувати як локальні.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1) збір і аналіз даних для створення і наповнення бази даних;
2) розробка автоматизованої web-системи обробки, зберігання та відображення даних кадастру;
3) розробка локальної системи збору даних.

Для вирішення першого завдання необхідно вирішити такі задачі:

1) розробка вимог до створення бази даних;
2) проведення аналізу і обґрунтування вибору типу бази даних;
3) проведення збору необхідних даних для наповнення бази даних;
4) проведення обробки зібраних даних, наповнення бази даних і вдосконалення її;

5) вирішення проблеми паралельного доступу і модифікації одних і тих же даних.

Враховуючи табличний характер інформації доцільно використовувати реляційну базу даних MySQL, розробку і підтримку якої здійснюється корпорація Oracle. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або віддалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера , що дозволяє включати MySQL в автономні програми.

Якщо користувач працює не з самою БД, а в локальному режимі, то немає необхідності відразу відправляти зміни в БД. Користувач на своєму комп'ютері робить всі необхідні правки, додає потрібні йому об'єкти і ці дані накопичуються. Після цього, якщо він вирішує що з'явилася необхідність завантажити ці дані в БД, він натискає на кнопку "Завантажити на сервер". У результаті цього виконується ініціалізація з'єднання з БД, проводиться авторизація та передані дані проходять перевірку на коректність. Якщо авторизація та перевірка на коректність завершуються успішно, то дані записуються в БД.

Бажано щоб всі дані, які користувач редагує, створює, завантажувалися на сервер хоча б раз на день. Це можна робити вручну або поставити автоматичне відправлення.

Якщо немає підключення до інтернету, то можна заздалегідь за допомогою даної програми скачати необхідну ділянку території і переглядати потім цю карту в локальному режимі.

Якщо одночасно декілька користувачів вивішать внести зміни в будь-яку таблицю. Для того що б ввести зміни вони повинні спочатку відкрити цю таблицю на зміну. Надсилається запит в БД, після обробки запиту на отримання таблиці, сервер відправляє таблицю користувачеві (тобто завантажується з бази даних у кеш браузера ), і розриває з'єднання. Тобто всі з'єднання виконуються тільки за такою схемою: відкриття з'єднання → відправлення запиту на сервер → обробка запиту → відправка відповіді користувачеві → закриття з'єднання. Після того як користувач відредагував таблицю, то відправка цієї таблиці виконується за аналогічною схемою. І якщо декілька користувачів одночасно вирішать відправити на сервер відредаговані таблиці, то по-замовчуванню в БД залишиться тільки та таблиця яку внесли пізніше за всіх. Але той користувач, який перший відкрив таблицю на редагування може поставити тимчасову заборону на редагування цієї таблиці іншим користувачам, і після остаточного редагування ця заборона знімається.

Структура бази даних Кадастру формувалася у вигляді семи взаємозв’язаних таблиць (Рис. 13.6).

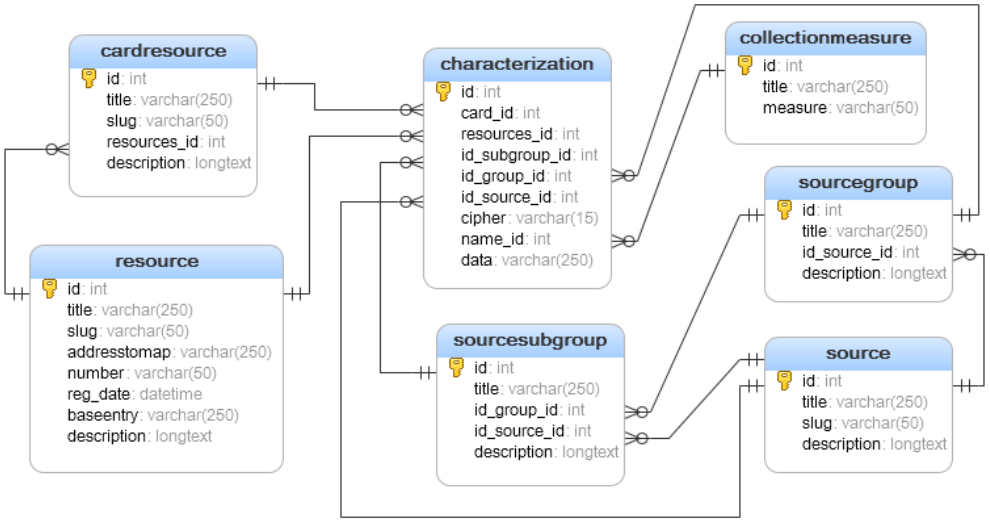



Рисунок 13.6 Структура бази даних

Принциповим для структури бази даних Кадастру є визначення абстрактного первинного елементу – об'єкту природних лікувальних ресурсів, який визначається унікальними географічними координатами. До об'єкту природних лікувальних ресурсів залучається загальна інформація відповідно типу природних лікувальних ресурсів, дані щодо кондицій, запасів, ліцензії на використання, адміністративне розташування, розташування за природними територіями курортів, земельними ділянками, дані щодо користувачів природних лікувальних ресурсів (рис. 13.7).

| Шифр | Назва показника | Дані |
|---|---|---|
| 1 | Родовище мінеральних вод | Маринівське «Аква Віта» |
| 2 | Ділянка родовища мінеральних вод | «Аква Віта» свр. № 4749 та № 4870 |
| 3 | Курортна територія | |
| 4 | Розташування за адміністративним підпорядкуванням | |
| 4.1 | назва населеного пункту | с. Маринівка |
| 4.2 | назва адміністративного району | Біляївський |
| 4.3 | назва області | Одеська |
| 5 | Геологічний індекс водоносного горизонту | N1S |
| 6 | Тип родовища | |
| 7 | Тип мінеральної води | хлоридна натрієва |
| 8 | Кондиційні показники | |
| 8.1 | мінералізація | |
| 8.1.1 | мінералізація (найменше значення) г/куб.дм | |
| 8.1.2 | мінералізація (найбільше значення) г/куб.дм | |
| 8.2 | основні іони | |
| 8.2.1 | гідрокарбонати | |
| 8.2.1.1 | гідрокарбонати (найменше значення) екв.% | |

Рисунок 13.7 Приклад таблиці

Вирішення другої задачі передбачає:

1) розробку автоматизовану web-систему обробки, зберігання та відображення даних кадастру;

2) реалізацію виведення в електронному та паперовому вигляді результатів аналізу і стандартних звітів.

Для вирішення цих завдань необхідно враховувати, що дана web-система має властивість розширюватися і змінятся з часом і для уніфікації розробки доцільно буде використовувати систему керування вмістом Drupal. Також для відображення об'єктів кадастру буде використовуватися картографічний сервіс Google Maps.

Drupal – популярна вільна модульна система керування вмістом (СКВ) з відкритим кодом, написана на мові програмування PHP.

Структура та потужна база модулів Drupal'у дозволяє порівняно швидко створювати потужні інтерактивні сайти.

До базового пакету системи, окрім модулів створення статичних сторінок (сторінок з постійною адресою) та нових статей входять модулі для організації блогів (електронних журналів користувачів), форумів (місць для інтернет-дискусій), «книг» (інформаційних добірок, праця над якими ведеться колективно), синдикації (імпорту новин з інших сайтів), модуль керування інформаційними блоками на сторінках, що полегшують керування їх виглядом, модуль керування меню.

Drupal має модульну архітектуру з компактним ядром, що надає API, до якого можуть звертатися модулі. Стандартний набір модулів включає такі функції, як новини стрічка, блог, форум, завантаження файлів, збирач новин, голосування, пошук тощо. Дизайн сайту змінюється також за допомогою спеціальних модулів – «тем оформлення» (Рис. 13.8).

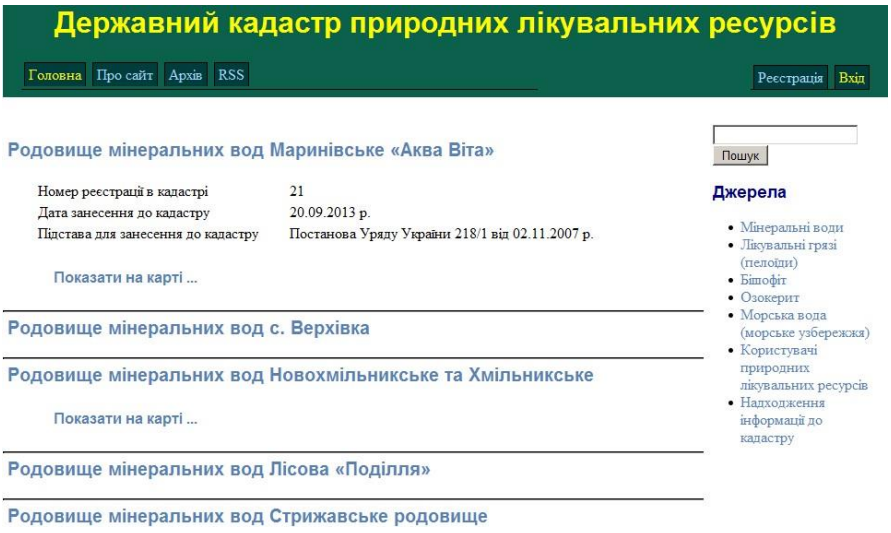



Рис. 13.8. Зовнішній вигляд web-системи

Google Maps – набір додатків, побудованих на основі безкоштовного картографічного сервісу і технологій, які надає компанія Google (Рис. 13.9).

Сервіс являє собою карту та супутникові знімки всього світу.

Третя задача яку необхідно вирішити це задача розробки локальної системи для збору даних кадастру.

Локальний додаток, буде містити частину бази даних, яку вибрав користувач, а також буде використовуватися для введення даних кадастру в випадку відсутності доступу до інтернету.

Локальний додаток буде написаний на мові програмування Java.

Java – це об’єктна-орієнтована мова програмування випущена компанією Sun Microsystems.

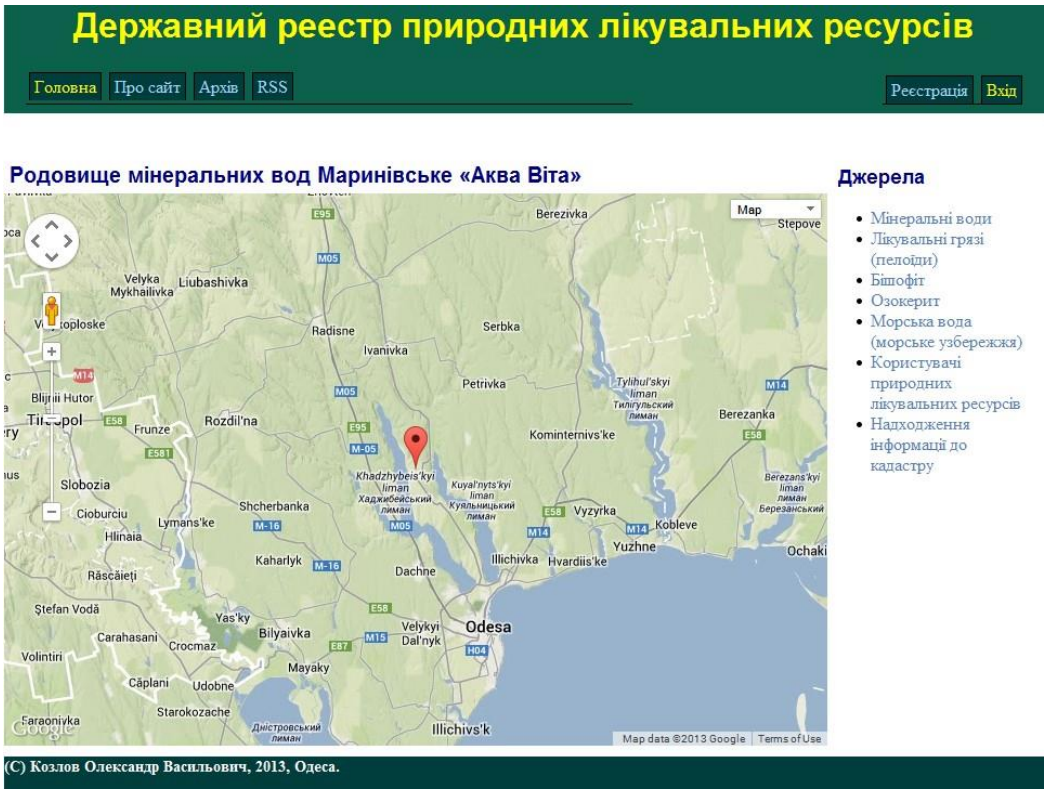



Рисунок 13.9 Приклад відображення об’єкту на карті

Вибір цієї мови обґрунтований тим що:

1) синтаксис мови простий та об’єктно-орієнтований;
2) незалежність від архітектури;
3) вона динамічна, інтерпретована та підтримує мультиопрацьовування.

Вирішення цих завдань дозволяють майже повністю досягти поставленої цілі.

***Наукова значимість*** проведених досліджень полягає в наступному:

1) запропонована архітектура геоінформаційної системи кадастру природних лікувальних ресурсів, яка поєднує властивості глобальних і локальних структур, за рахунок гнучкої взаємодії розподілених локальних абонентів з центральною базою даних через автоматизовану систему зберігання, обробки і відображення даних кадастру;

2) знижена невизначеність вмісту центральної бази даних для розподілених споживачів за рахунок істотного скорочення часу внесення коректив.

***Висновки.***

В результаті виконання даної роботи:

1) розроблена, проаналізована, наповнена і оптимізована база даних кадастру;

2) впроваджена автоматизована web-система обробки, зберігання та відображення даних кадастру;

3) реалізоване виведення в електронному та паперовому вигляді результатів аналізу і стандартних звітів.

Остання задача створення локальної системи для збору даних кадастру на даний час знаходиться у розробці.

Розроблена інформаційна система має такі можливості:

1) Відкритий доступ, тобто для доступу до кадастру потрібен тільки комп’ютер і доступ в інтернет.
2) Добре масштабована (1 до 20м).
3) Дуже висока швидкість рендерінгу.
4) Зручна для роботи існуючих систем (комп'ютерів), не вимоглива до ресурсів.
5) Можливість фільтрації даних.
6) Різний аналіз даних.
7) Відображення короткої інформації про об'єкт.
8) Отримання інформації по заданих параметрах.
9) Можливість додавання нового об'єкта і зміна існуючого.
10) Можливість завантаження GPS треків.
11) Висока продуктивність ГІС системи.
12) Робота з картою в реальному часі без перезавантаження сторінки.
13) Можливість включення і виключення відображення кожного тематичного шару.
14) Можливість роботи в оффлайн режимі.

# 14 ІНФОРМАЦІЙНІ СИСТЕМИ ЕКОЛОГІЧНО НЕБЕЗПЕЧНИХ ОБ'ЄКТІВ

Одним із аспектів екологічної безпеки мегаполісів є їх техногенна безпека, яка визначається станом потенційно небезпечних об'єктів (ПНО), таких як автозаправні станції (АЗС) та комплекси, підприємства хімічної промисловості тощо. Потенційно небезпечним об'єктом вважається об'єкт, на якому використовуються, виготовляються, переробляються, зберігаються чи транспортуються небезпечні речовини, біологічні препарати тощо. Особливість діяльності ПНО призводить до техногенних ризиків та за певних обставин створює реальну загрозу розвитку сценарію аварії (матеріали з дисертаційної роботи Іванова О.В. Моделі та методи аналізу зон ризику потенційно небезпечних об'єктів в геоінформаційних системах).

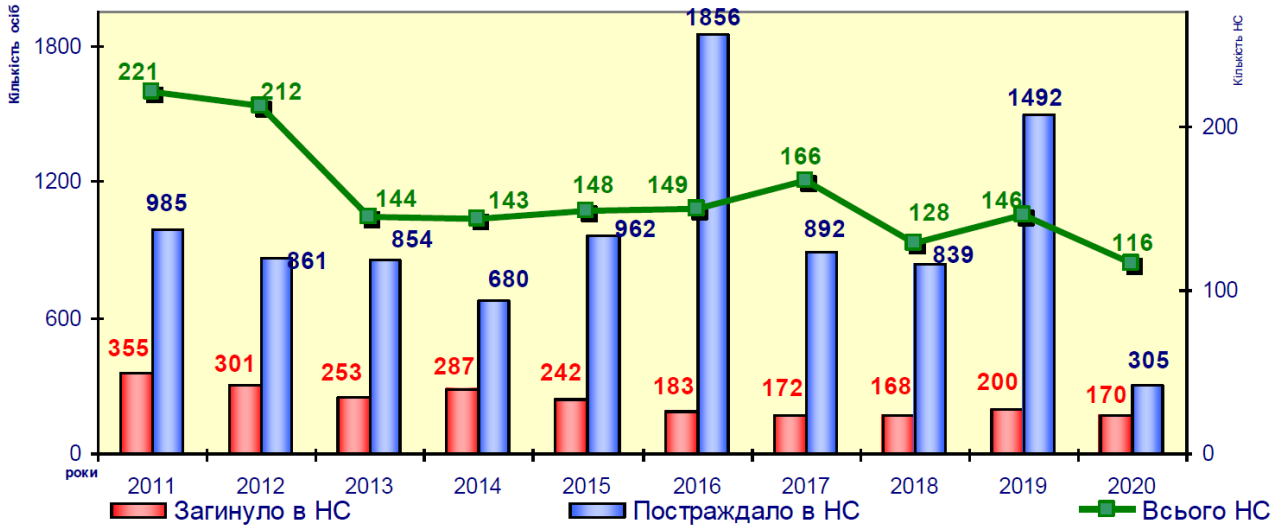



Рисунок 14.1 Динаміка виникнення НС в Україні (матеріали з дисертаційної роботи)

Аналіз небезпеки та ризику аварій на потенційно небезпечному об'єкті включає такі основні етапи:

- постановка завдання аналізу небезпеки та оцінки ризику;
- аналіз небезпеки та умов виникнення аварій;
- оцінка ризику (ймовірності) виникнення аварій;
- аналіз умов і оцінка ймовірності розвитку аварій;
- визначення масштабів наслідків;
- оцінка ймовірності наслідків аварій;
- оцінка прийнятності ризику та прийняття рішень щодо зменшення ризику.

Відсутність вітчизняних інформаційних систем Рис. 14.2) зумовлює актуальність задачі створення подібних систем екологічної направленості.

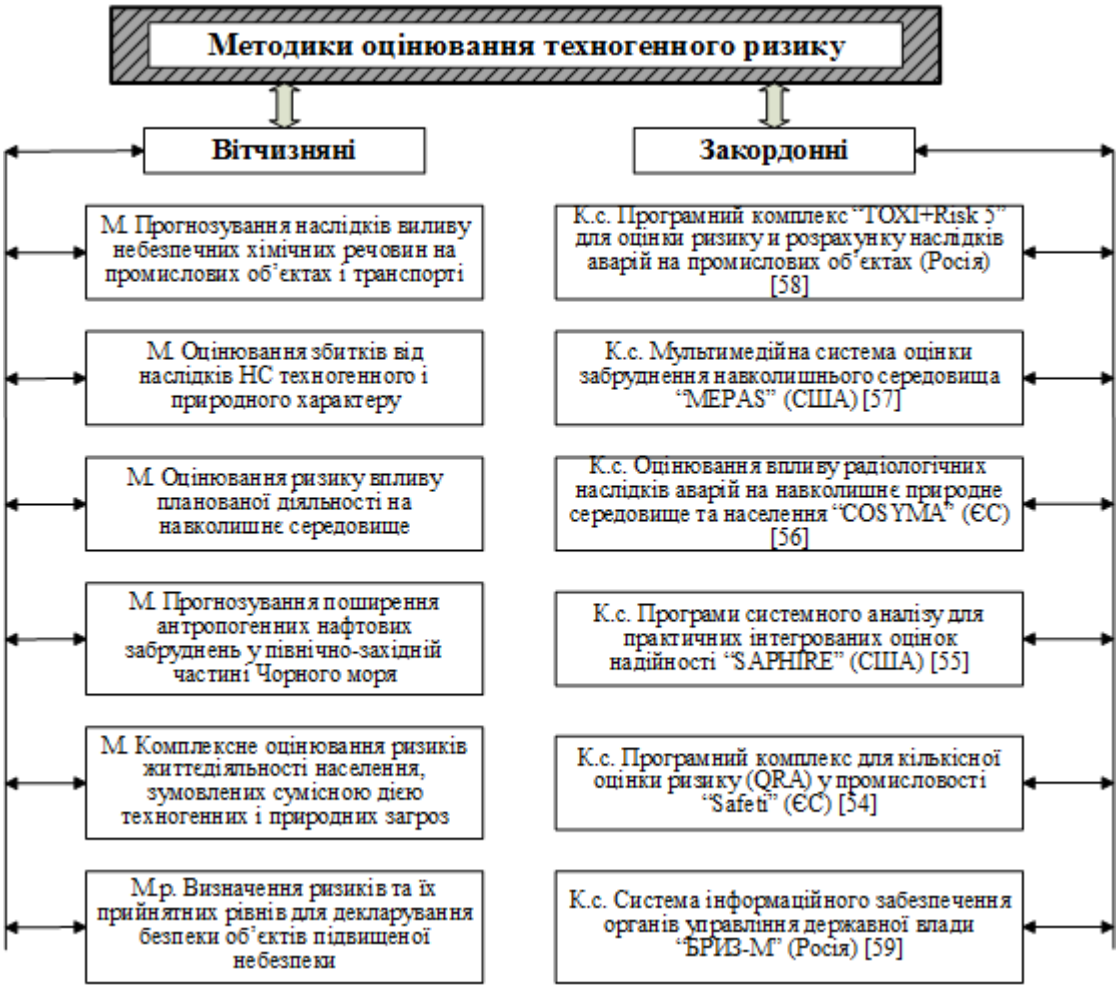



Рисунок 14.2 Основні вітчизняні та закордонні методики і комп'ютерні системи оцінювання техногенного ризику (М. – методика; М.р. – методичні рекомендації; К.с. – комп'ютерна система)

Шляхом розв'язання цих проблем є створення власного програмного забезпечення, що дозволить проводити розрахунки і моделювання НС на ПНО за вітчизняними затвердженими методиками, а також візуалізація отриманих даних засобами вільних геоінформаційних систем. У подальшому результати досліджень можуть бути використані для створення повноцінних комп'ютерних систем оцінювання техногенного ризику широкого кола питань.

Метою даної роботи є  зниження часу на визначення сценарію розвитку аварії на потенційно небезпечному об'єкті за рахунок розробки моделі та методів аналізу та візуалізації зон техногенного ризику автозаправної станції (АЗС) в геоінформаційних системах. Для досягнення поставленої мети сформульовані  такі завдання:

– проаналізовати методи визначення сценарію розвитку аварії на потенційно небезпечному об’єкті та можливості існуючих геоінформаційних систем щодо аналізу зон техногенного ризику, обґрунтовано мету та задачі дослідження;

– розробити моделі геоданих автозаправної станції як потенційно небезпечного об’єкта та зон техногенного ризику від ударної хвилі, пожежі проливу нафтопродуктів і «вогняної кулі»;

– розробити методи визначення та візуалізації геоданих зон техногенного ризику при розвитку небезпечної ситуації (аварії) за заданим сценарієм;

– розробити підсистему аналізу та візуалізації геоданих зон техногенного ризику.

При розробці цієї моделі виходили із наступних двох положень:

1. Дана модель АЗС має відповідати вимогам нормативно-затверджених вітчизняних методів розрахунку критеріїв пожежно- та вибухопожежної безпеки зовнішніх установок, які дозволяють проводити розрахунок зон ризику при виникненні на АЗС однієї із трьох НС: вибух пароповітряної суміші (ППС) з виникненням ударної хвилі; пожежа проливу нафтопродуктів; виникнення «вогняної кулі». Крім того, при розрахунку розмірів зон ризику мають бути враховані типові значення тиску у фронті ударної хвилі й інтенсивності теплового випромінювання пожежі з точки зору їх впливу на людей і навколишні об’єкти (будівлі, устаткування, автомобілі тощо).

2. Для візуалізації зон ризику від АЗС як типового ПНО засобами геоінформаційних систем дані про АЗС мають бути представлені у вигляді геоданих, а для побудови зон ризику навколо них засобом буферизації має бути заданий радіус зони ризику (або буферної зони у термінах ГІС), що має відповідати попередньому положенню.

Виділяють 2 типові сценарії розвитку аварії на АЗС.

Перший сценарій передбачає повне зруйнування ємності з повним вивільненням пожежовибухонебезпечної речовини, яка у ній зберігається. Причинами зруйнування ємності можуть бути різні ініціюючі події, викликані як внутрішніми, так і зовнішніми факторами, наприклад землетрус та зрушення земної поверхні, падіння літака та інших летальних апаратів, диверсії, терористичні акти, тепловий удар ті гідравлічний розрив.

Другий сценарій передбачає локальне зруйнування установок із ЛЗР чи ГР. Розрахунок проводиться для трьох варіантів:

– пожежа-спалах при локальному виході продукту із ємності;
– пожежа проливу ЛЗР чи ГР;
– «вогняна куля» при розриві ємності з речовиною під тиском.

При потраплянні у межі «вогняної кулі» чи проливі сусідніх резервуарів останні з вірогідністю 60 % вибухають у результаті ефекту «BLEVE», утворення якого відбувається за наступною схемою:

- охоплення полум'ям резервуару, підвищення тиску всередині резервуару та нагрів металу з втратою його міцності;
- розрив оболонки резервуару, викид речовини та скипання викинутої речовини;
- вибух парів рідини, яка скипіла, із запалюванням та утворенням «вогняної кулі».

Модель розвитку надзвичайної ситуації представлена на рис. 14.3.


Рисунок 14.3 Аналіз розвитку НС на АЗС внаслідок руйнування ємності за допомогою "дерева подій"

При розробці концептуальної геопросторової моделі АЗС враховані наступні складові для подальшого моделювання та візуалізації зон ризику засобами ГІС:

1) атрибутивна складова цифрової моделі АЗС повинна містити набір даних для подальшого проведення чисельного моделювання розвитку несприятливих ситуацій на АЗС за затвердженими законодавством методиками;

2) просторова складова цифрової моделі АЗС має бути представлена згідно вимогам відповідної ГІС;

3) динамічна складова цифрової моделі АЗС має враховувати особливості сценарію розвитку НС на основі атрибутивної та просторової інформації, а також обраної моделі розвитку НС на основі відповідної методики розрахунку.

Схему взаємозв'язку складових розробленої концептуальної геопросторової моделі АЗС показано на рис. 14.4.

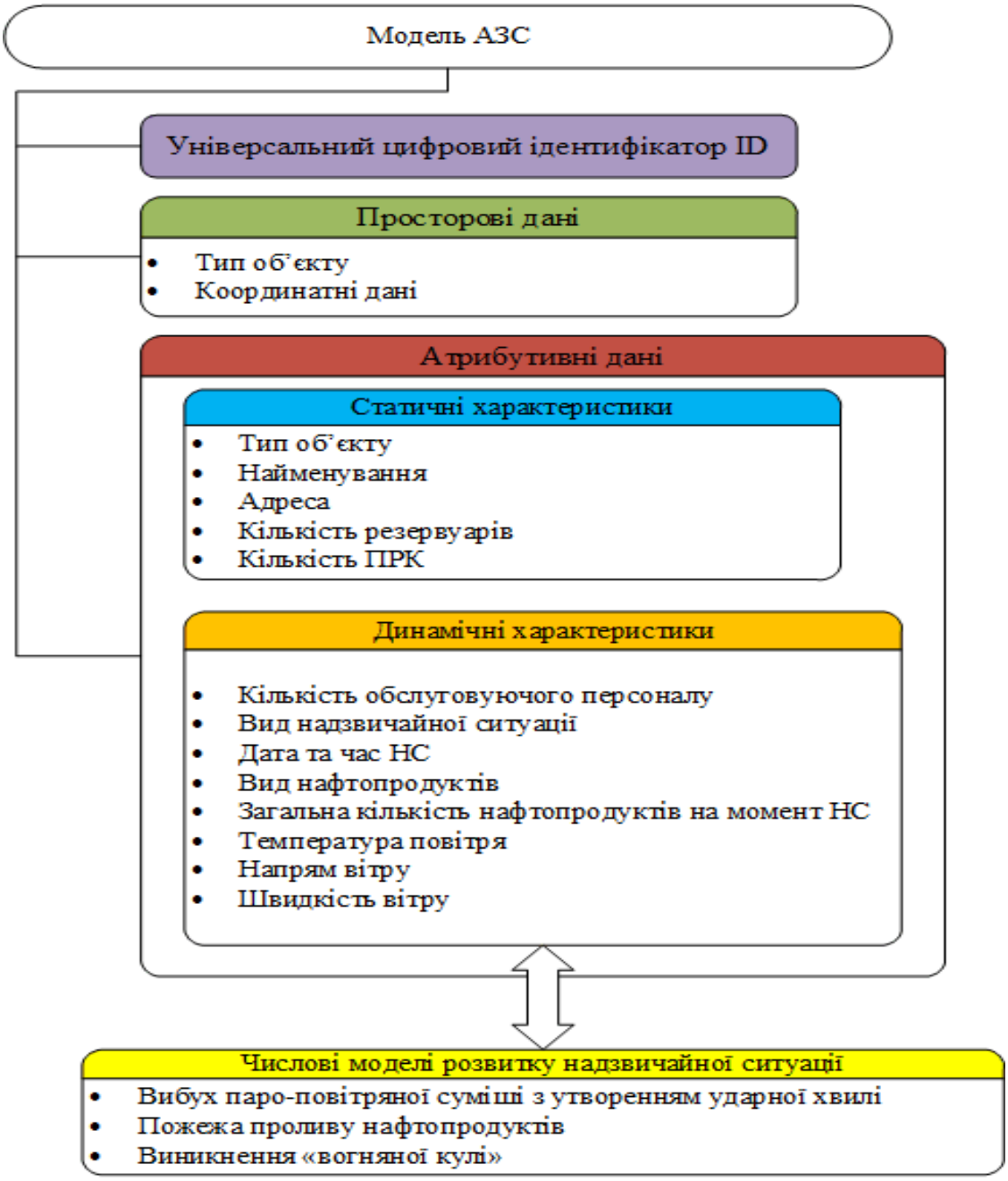



Рисунок 14.4 Схема взаємозв'язку складових концептуальної геопросторової моделі АЗС

В якості атрибутивних характеристик досліджуваного об'єкта пропонуються наступні (їх перелік не вичерпується наведеними і може бути збільшений відповідно до потреб моделювання):

– тип об'єкту (АЗС);
– найменування (назва фірми-власника);
– адреса (юридична адреса);
– кількість резервуарів із зазначенням їх типу (наземний, підземний);

– кількість паливно-роздавальних колонок (ПРК);
– кількість обслуговуючого персоналу, чол;

вид надзвичайної ситуації (НС) (сценарій несприятливої події, що обирається для моделювання);

– дата та час НС;
– вид нафтопродуктів (із зазначенням, у якому із резервуарів вони зберігається);
– кількість нафтопродуктів (із зазначенням їхньої кількості на момент НС у конкретних резервуарах);
– температура навколишнього повітря, °С;
– напрям вітру;
– швидкість вітру у м/с.

На основі розробленої моделі геоданих автозаправної станції як потенційно небезпечного об'єкта та моделей геоданих зон ризику, що виникають на АЗС, розроблено метод визначення геоданих зон техногенного ризику при розвитку небезпечної ситуації (аварії) за заданим сценарієм, які потім будуть використовуватись у ГІС для візуалізації.

Визначені геодані зон ризику від ударної хвилі і проливу нафтопродуктів (рис. 14.5, Рис. 14.6).

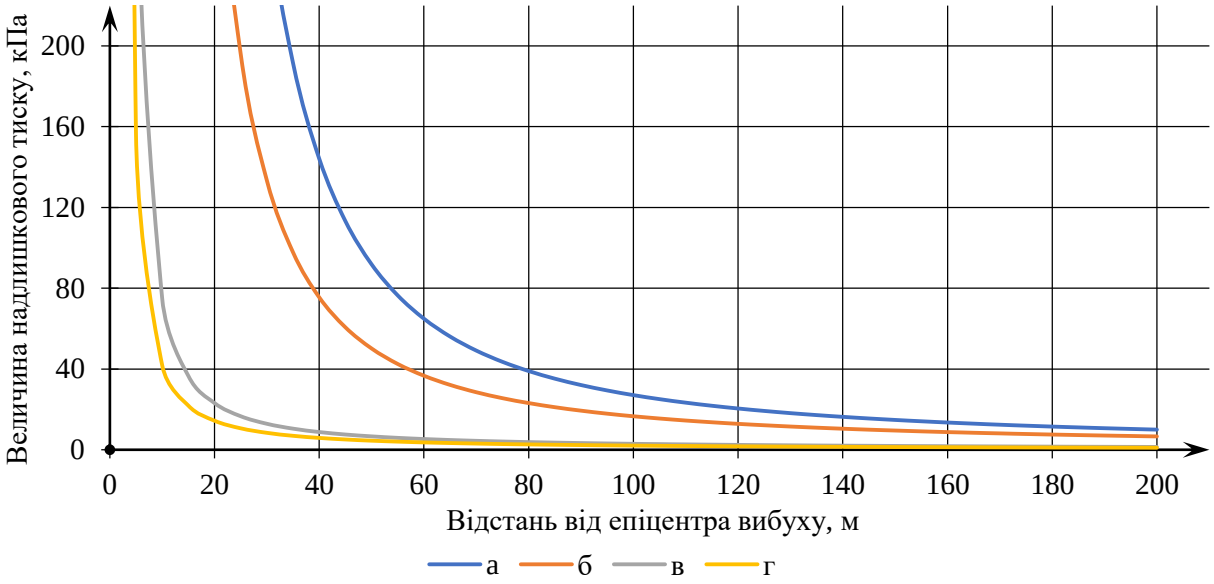



Рисунок 14.5 Графіки зміни величини надлишкового тиску у фронті ударної хвилі залежно від відстані від епіцентру вибуху: а, в – для резервуарів з бензином ємністю 40 і 15 м$^3$ відповідно; б, г – для резервуарів з дизельним паливом ємністю 40 і 15 м$^3$ відповідно

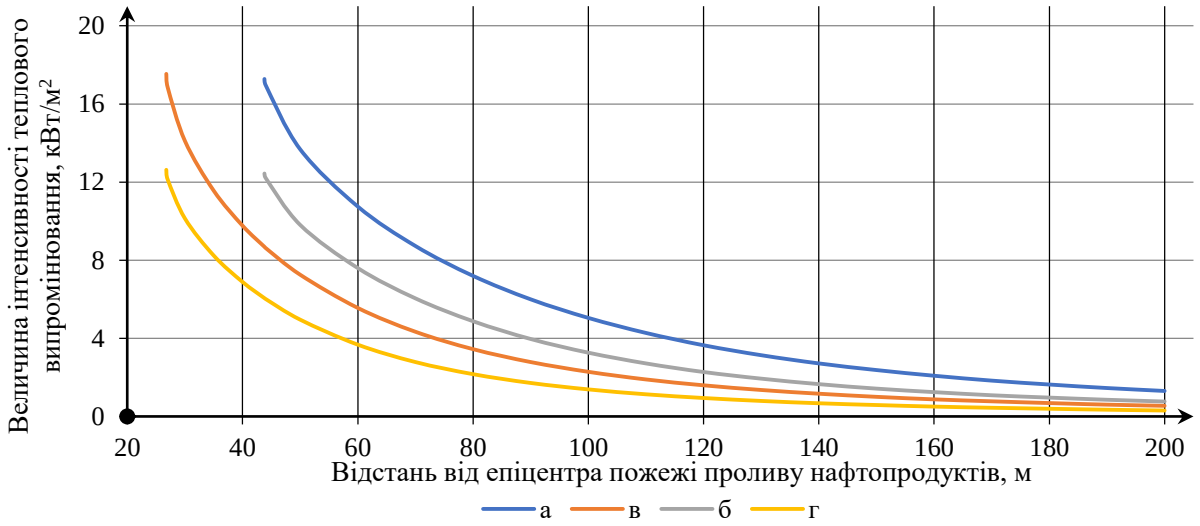



Рисунок 14.6 Графіки зміни величини інтенсивності теплового випромінювання залежно від відстані від епіцентру пожежі: а, в – для резервуарів з бензином ємністю 40 і 15 м$^3$ відповідно; б, г – для резервуарів з дизельним паливом ємністю 40 і 15 м$^3$ відповідно

Візуалізація мережі АЗС на карті міста реалізовано за допомогою плагінів *QGIS* (Рис. 14. 7, Рис. 14.8).

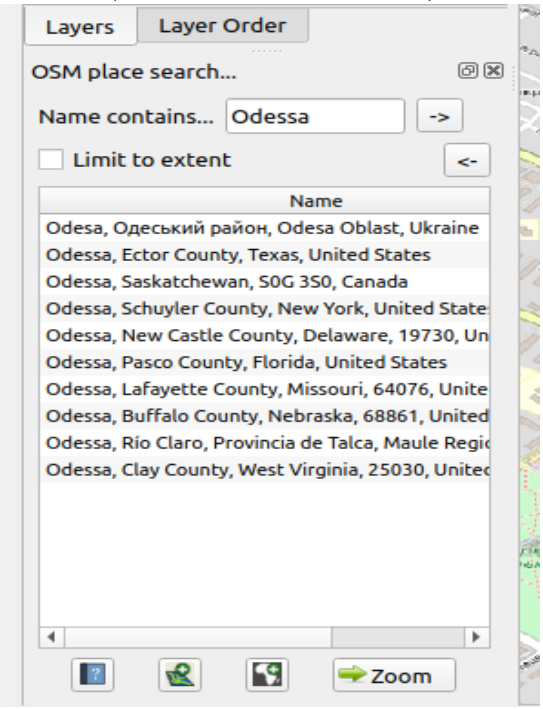




Рисунок 14.7 Результати налаштувань плагінів *QGIS* для візуалізації мережі АЗС, як потенційно-небезпечних об’єктів (а –діалогове вікно плагіну *OSM Place Search*; б – текст скрипта запиту на сервер openstreetmap)

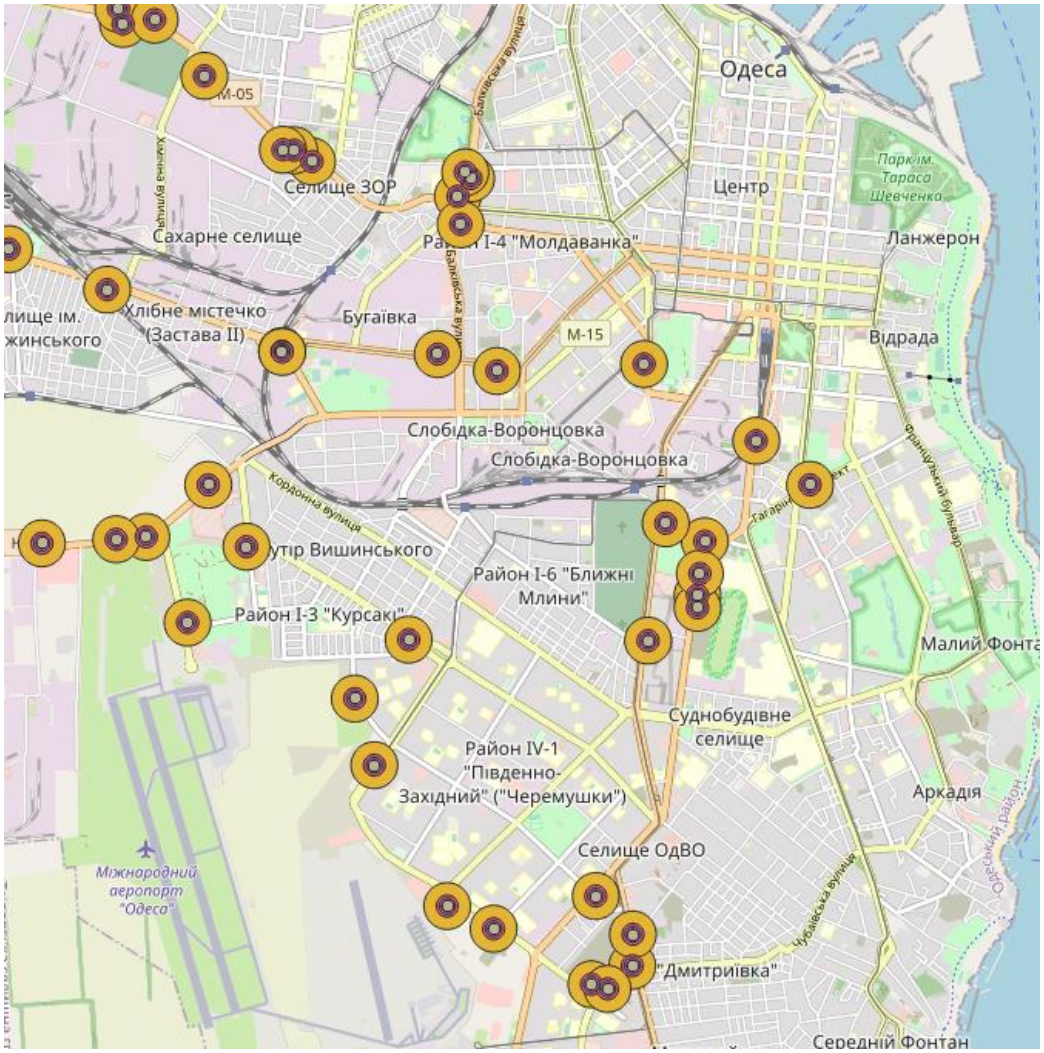



Рисунок 14.8 Результати візуалізації геоданих моделі зон техногенного ризику, що виникає від ударної хвилі $RZM_{SW}$ на карті міста у підсистемі *QGIS*

На основі аналізу побудованої карти (рис. 3.9) можна зауважити наступне для АЗС з резервуарами з бензином ємністю 40 м$^3$:

– на деяких АЗС не виконуються норми протипожежних відстаней до так званих об’єктів «турботи» (підприємств, організацій, селітебних зон тощо);

– деякі АЗС розміщені не тільки в селітебних зонах, але і поблизу автомагістралей із значним потоком автомобілів, що особливо у пікові години збільшує негативні наслідки аварії;

– деякі АЗС розміщено одна поблизу одної, наприклад, традиційна та газова АЗС, що у випадку аварії може спричинити розвиток сценарію за каскадним ефектом «доміно» – приклад для одного з районів міста зображено на рис. 14.9 (зони ризику на карті перекриваються).

Рисунок 14.9 Візуалізація зон техногенного ризику від ударної хвилі за умов максимальної місткості (40 м$^3$) резервуарів із бензином на одному із районів

За умови мінімального заповнення резервуарів АЗС нафтопродуктами (15 м$^3$) зони ризику у більшості випадків є локальними (на території АЗС), проте можливим є вплив на об’єкти «турботи» в сусідніх селітебних зонах.

Як видно із прикладу візуалізації геоданих моделі зон техногенного ризику від ударної хвилі $RZM_{SW}$ у підсистемі QGIS у великому масштабі (рис. 14.10), у зону середніх руйнувань (тиск від 30 до 10 кПа) потрапляє частина селітебної території (а саме, 2 багатоповерхові будинки), газорозподільча станція, заклад громадського харчування. При цьому у житлових будинках частково деформується більшість несучих конструкцій, з’являються тріщини в зовнішніх стінах і провали в окремих місцях, при цьому другорядні та частина несучих конструкцій можуть бути зруйновані повністю. На комунально-енергетичній мережі пошкоджуються технологічні трубопроводи, що може викликати ефект «доміно» на газорозподільчій станції. У людей, які потрапляють у цю зону, можуть виникнути легкі травми (контузії, вивихи кінцівок, тимчасова втрата слуху).

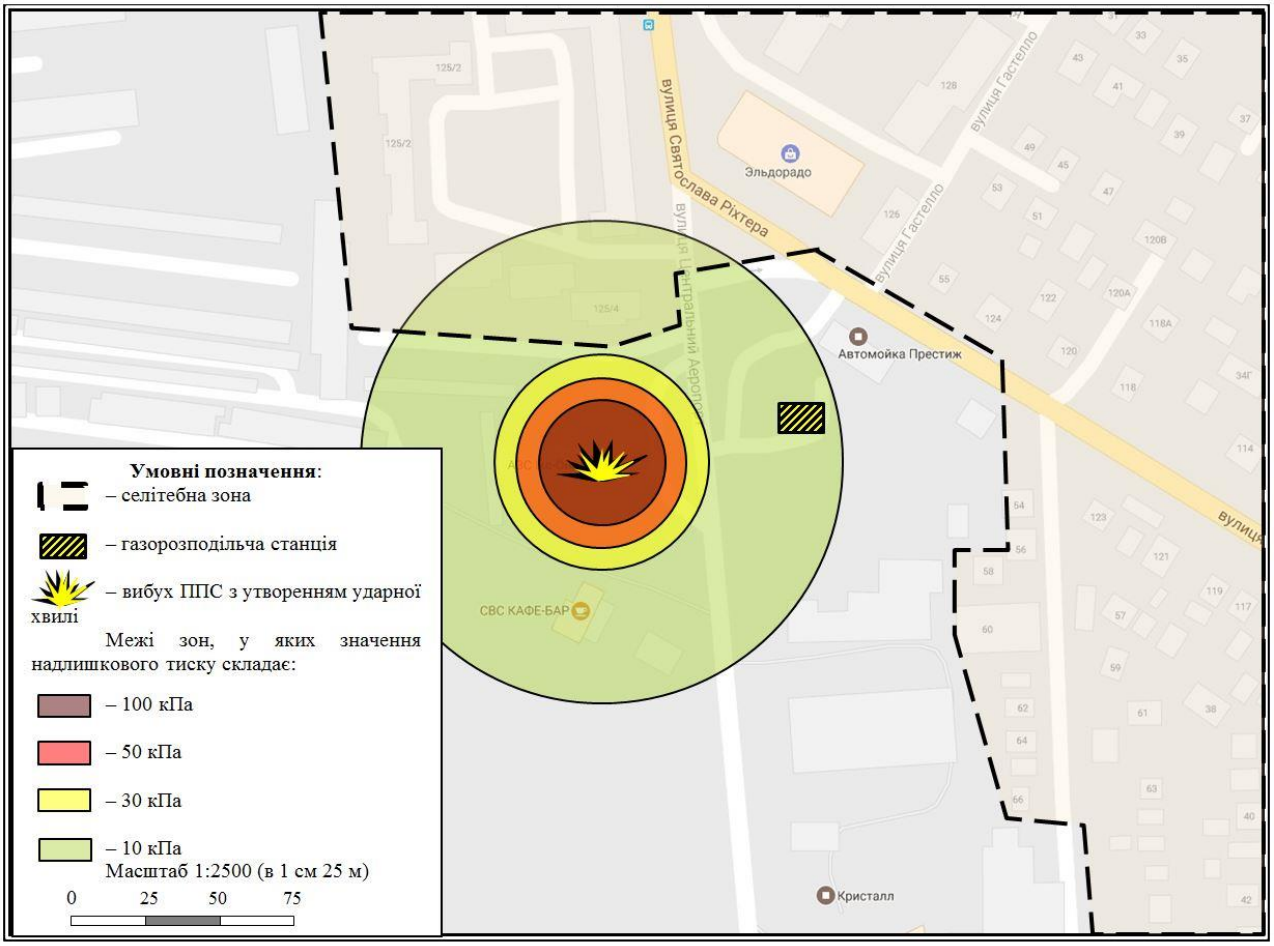



Рисунок 14.10 Результати візуалізації зон техногенного ризику від ударної хвилі на прикладі однієї з АЗС

Варто також відмітити, що практично повністю від ударної хвилі може постраждати не лише АЗС, а й розташована поруч автомийка (20 м), а також сильних руйнувань зазнає гаражний кооператив. Також можна побачити, що значна частина автостради підпадає під зону дії ударної хвилі, що може викликати значні людські жертви, коли у годину пік поблизу перехрестя збирається велика кількість автомобілів. А кількість людських жертв збільшується не лише за рахунок персоналу АЗС, а і персоналу автомийки.

При наявності даних про вітер (особливо актуально для пожежі проливу нафтопродуктів, зважаючи на можливу довготривалість цього сценарію, на відміну від вибуху ППС з утворенням ударної хвилі та «вогняної кулі»), ми можемо скористатись плагіном *Shape Tools* і використати їх для відображення зон ризику з поправкою на вітер (приклад реалізації зображено на рис. 14.11).

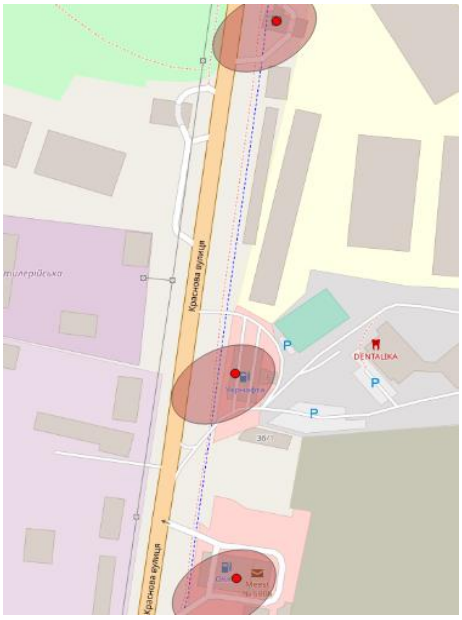



Рисунок 14.11 Результати візуалізації геоданих моделі зон техногенного ризику *RZM* від АЗС на карті з поправкою на вітер

Загальна структура системи визначення та візуалізації зон техногенного ризику АЗС як ПНО, яку побудовано на основі геоінформаційної системи з відкритим кодом *QGIS* складається з наступних складових: відкритого картографічного сервісу *OpenStreetMap* для отримання початкової картографічної інформації (або просторових даних), метео сервісу *OpenWeather* з відкритим *API* для отримання часових динамічних даних про метеоумови, бази геоданих (*PostGIS*), яка реалізована у вигляді розширення до *PostgreSQL*, веб-браузера, серверної Веб ГІС (*NextGIS*). Система є автоматизованою та працює за участю користувача-експерта (рис. 14.12). Метою спеціалізованої ГІС є визначення (розрахунок) та візуалізація зон техногенного ризику АЗС як ПНО при виникненні НС на них.

Користувач підсистеми має змогу працювати із геоінформаційною системою *QGIS*, базою даних *PostGIS* та веб-браузером. База даних *PostGIS* використовується для збереження даних, отриманих при роботі з спеціалізованою ГІС.

Інформаційна частина (*OpenStreetMap, OpenWeather*) надає вхідні дані до геоінформаційної системи *QGIS*. ГІС *QGIS* використовується як основний елемент визначення (розрахунку) зон техногенного ризику (за допомогою спеціально розробленого плагіну *PHO*) та їх візуалізації на карті населеного пункту. Серверна частина (реалізована у вигляді серверної Веб ГІС *NextGIS*) використовується для відображення результатів візуалізації на веб-сайтах.

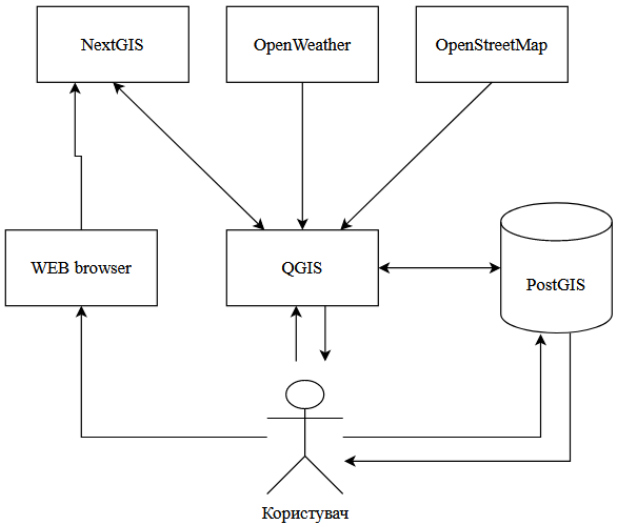



Рисунок 14.12 Структура спеціалізованої інформаційної системи

Інформаційна частина розробленої підсистеми складається з сервісів *OpenStreetMap* та *OpenWeather*. *OpenStreetMap* – це відкритий проект, спрямований на збір, збереження та розповсюдження загальнодоступних геопросторових даних, створення інструментів для роботи з ними силами спільноти волонтерів.

Базові інструменти *OSM* оперують даними у форматі XML, який описує набір екземплярів елементів (вузли, відрізки та зв'язки). У нашому випадку ми генеруємо мапи локально за допомогою *QGIS* та плагінів.

Отримання даних відбувається за допомогою API.

Отримання часових динамічних даних про метеоумови виконується за допомогою сервісу *OpenWeather*.

*OpenWeather* – онлайн сервіс, який надає API для доступу до даних про поточну погоду, прогнози та історичні дані. Як джерело даних використовуються офіційні метеорологічні служби, дані з метеостанцій аеропортів та дані з приватних метеостанцій.

# 15 ІНФОРМАЦІЙНІ МЕТЕОРОЛОГІЧНІ ПІДСИСТЕМИ

Інформативні системи функціонують у реальних умовах довкілля, тому зовнішні чинники безпосередньо впливають на поведінку системи. Прикладами таких впливів може бути вплив вітру, температури та вологості повітря на поширення пожежі, напрямки та концентрації шкідливих викидів від промислових підприємств та автомобільного транспорту на розташування та якість лікувальної лиманної рапи та пелоїду тощо. Локальні гідрометеорологічні системи є необхідним компонентом інформаційної системи, оскільки вимагають показань у реальному часі основних складових: швидкості та напрямки вітру, вологості, наявності опадів, тиску.

Функціонально метеорологічна система не обмежується наведеними застосуваннями. В залежності від компонування вона може бути застосована для спостереження різноманітних показників:

• Інформування водіїв про погодну обстановку на автомобільних шляхах. Такого роду метеостанція, окрім загальних вимірів, вимірюють температуру поверхні дорожнього покриття та на глибині 30 сантиметрів під ним. Зазвичай обладнані таблом, яке в реальному часі інформує водіїв, та GPRS модулями для передачі даних в інформаційні центри.

• Збір кліматичних даних у лісах, попереджують про можливу пожежу. Зазвичай такі прилади працюють від акумуляторів та вимірюють вологість дерева та ґрунту, температуру на різних рівнях висотності лісів. Дані вимірів системи датчиків обробляються й моделюється карта пожежної активності, що запобігає можливій масштабній пожежі.

• Спостереження над станом погоди океанів, морів, річок, озер та боліт. Станції даного типу "слідкують" за метеорологічними та гідрологічними показниками, розташовуються на суші, на морських плаваючих станціях та річках, озерах і болотах.

• Інформування про стан метеорологічних та інших показників вдома. Домашні метеостанції вимірюють менше показників, але їх перевагами являються великий вибір на ринку, різноманітність вбудованих датчиків, можливість з дому власноруч вести спостереження про стан тих чи інших показників, компактність та доступність.

Також, додатково обладнавши метеостанцію спеціальним датчиком забрудненості повітря, можна з легкістю спостерігати за вмістом та кількістю шкідливих і небезпечних газів в повітрі таких як: $NH_3$, $NO_x$, пари алкоголю, бензину, диму, $CO_2$ тощо. Це значно розширює об'єм корисної інформації не тільки для рядового користувача, але й для цілих населених пунктів. Можливе застосування й для аграрної промисловості. Така важлива галузь потребує відповідного рівня контролю показників температури, вологості повітря й грунту, освітленості поверхні протягом певного періоду часу, УФ-

випромінювання. Основною ціллю метеорологічних спостережень й самої метеорології являється виявлення та картування різноманітних атмосферних явищ, а згодом складання прогнозу майбутньої погоди опираючись на дані попередніх кліматичних подій.

Вимірювані фізичні величини Атмосферний тиск, температура повітря, відносна вологість повітря, (концентрація аміаку, оксидів азоту, парів алкоголю, бензину, диму та $CO_2$), (швидкість та напрям вітру), (кількість опадів), (освітленість видимого спектру та УФ променів), (вологість та температура грунту) кожна пара дужок відповідає окремому датчику.

Для обробки інформації з датчиків, для керування ними та пристроєм, який буде відповідати за безпровідну передачу даних, та для додаткових користувацьких функцій потрібно електронно-обчислювальну систему. Для таких неважких задач ідеально підходить мікроконтролер. Найпростіша структурна схема, яка дозволяє реалізувати функціонал пристрою, який конструюється, повинна містити в собі наступні вузли та компоненти:

- Блок живлення
- Датчики
- Мікроконтролер
- GSM-модуль.

Основними показниками для системи є швидкість і напрям вітру. Важливою характеристикою вітру є його мінливість у часі та просторі. Мінливість у часі обумовлює необхідність визначення кількох параметрів, що належать до вітру: середня, мінімальна та максимальна величини.

Вимірювання параметрів вітру здійснюється за допомогою анемометрів. Найбільш поширеними з анемометрів з крильчатками є чашкові анемометри або анемометри з млином, швидкість обертання яких синхронна зі швидкістю вітру.

Крім того, існують також статичні плівкові термоанемометри та ультразвукові датчики. У них відсутні рухомі механічні частини і вони є технічно більш складними і можуть позбуватися льоду краще, ніж більшість датчиків, що обертаються. Ультразвукові датчики також мають коротку константу часу і здатні робити велику кількість вимірювань за секунду. Однак при цьому важливо інтегрувати дані цих вимірювань за 3-секундний період для пікових значень швидкості та напряму.

Основним приладом, що вимірює характеристики, вітру є АНЕМОРУМБОМЕТР М63М-1. М63М-1 - призначений для дистанційних вимірювань миттєвої, максимальної та середньої швидкостей та напряму вітру у стаціонарних умовах. До складу приладу входять датчик вітру, пульт цифрової обробки та відображення результатів вимірювань, кабель, блок живлення.

Технічні характеристики.

Діапазон вимірів:

Миттєву швидкість вітру 1,5 до 60 м/с; максимальну швидкість від 3 до 60 м/с; середньої швидкості від 12 до 40 м/с; напрямки вітру від 0 до 3600;

Основна похибка при вимірі швидкості вітру трохи більше +(0,5+0,05V) м/с, де V – вимірювана швидкість вітру;

Основна похибка при вимірі напряму +100;

Поріг чутливості датчика вітру трохи більше: По швидкості вітру 0,8 м/с. у напрямку вітру 1,2 м/с;

Принцип дії приладу ґрунтується на залежності

1) частоти (наслідування) імпульсів, що виробляються датчиком, від швидкості вітру.

2) зсуву фази між двома групами імпульсів, що виробляються датчиком від напрямку вітру.

Під впливом вітру генераторами датчика виробляються 3 групи імпульсів напруги рівної частоти, що залежить тільки від швидкості вітру.

Первинний перетворювач швидкості – повітряний гвинт. Значення поточної (миттєвої) швидкості виробляється шляхом перетворення генераторами датчика імпульсів групи ОС. Напруга з його виході пропорційно частоті вихідних імпульсів, вимірюється двошкальным міліамперметром. Значення середньої швидкості виробляється в результаті підрахунку імпульсів групи ВП, що надходять від датчика за 10 або 2 хв. Тимчасовий інтервал визначається годинниковим механізмом. Максимальна швидкість вітру запам'ятовується пасивною стрілкою.

Первинний перетворювач напряму – флюгарка. Значення напряму вітру виробляється генераторами датчика, з груп: ОС, СД, ОП. Імпульси групи ВП надходять на встановлювальний вхід тригера та повертають його у вихідне положення. Необхідність використання трьох груп визначається стрибкоподібною зміною зсуву фаз груп ОС, ВП при переході флюгарки через 0 ($360^o$), а групи ЦД, ВП при переході через $180^o$ в результаті підрахунку числа імпульсів групи ВП, що надходять від датчика за інтервал зосередження швидкості через масштабний дільник частоти на вхід механічного лічильника імпульсів. Масштабний дільник служить узгодження коефіцієнта перетворення датчика з прийнятим масштабом вихідний величини.

АНЕМОРУМБОМЕТР М63М1 З ВИХОДОМ НА ПК Призначений для дистанційних вимірювань значень миттєвої, максимальної та середньої швидкостей та напряму вітру. Застосовується у системах збирання метеорологічної інформації, диспетчерських аеродромних службах та інших

галузях народного господарства. До складу приладу входять датчик вітру, пульт цифрової обробки та індикації результатів вимірювань, перетворювач інтерфейсів RS-485 у RS-232, блок живлення, з'єднувальний кабель довжиною 100 м, програмне забезпечення. Для передачі в персональний комп'ютер анеморумбометр М63М-1 має симетричний інтерфейс RS-485 і комплектується перетворювачем інтерфейсів RS-485 в RS-232. Використання інтерфейсу RS-485 дозволяє здійснювати передачу даних лініями зв'язку між анеморумбометром М63М-1 і перетворювачем інтерфейсів RS485 в RS232 до 1200 метрів, між перетворювачем інтерфейсів і персональним комп'ютером до 5 метрів. Лінія зв'язку між датчиком вітру та анеморумбометром може досягати 300 метрів. Програма дозволяє одночасно обробляти дані від одного до восьми анеморумбометрів М63М-1.

### *Типи метеостанцій*

Метеостанція, призначена для локального використання, є нічим іншим, як приладом настінного або настільного типу, здатним визначати те, в якому стані перебуває мікроклімат усередині приміщення, і які погодні умови спостерігаються зовні. На сьогоднішній день існують різні види метеостанцій, і кожна з них є практичною через виконання великої кількості функцій.

1. Можливість визначення температурної позначки, а також вологості повітряної маси безпосередньо всередині та зовні.
2. Контроль мікроклімату не більше приміщення.
3. Упорядкування прогнозу погоди на добу вперед і навіть більше.
4. Додаткові функції у вигляді годинника, календаря, будильника.
5. Визначення параметрів тиску.
6. З'ясування показників щодо опадів (інтенсивності, обсягу) та вітру (швидкості, напрямку).
7. Показ відомостей про час, коли починається захід сонця і закінчується світанок.

### *Основні види метеостанцій*

Види метеостанцій численні, і перший варіант класифікації має на увазі існування аналогових та цифрових рішень. У першому випадку йдеться про комплекс механічних елементів з вимірювачами та класичною корпусною частиною. У другій ситуації станції є приладами портативного типу, які фіксують інформацію за допомогою використання датчиків в електронному вигляді.

Якщо говорити про такий параметр, як типи представлених на ринку метеостанцій, можна виділити такі напрямки.

1. Цифрове рішення із барометром. Крім зазначеного елемента, до складу також входить анемометр, флюгер, USB вихід та кілька допоміжних опцій. Стрілецький пристрій оснащений двома стрілками, одна визначає тиск, а другу користувач переміщає у становище минулого показника. Рівень точності середній.

2. Термометр. Він виступає як незначний за розміром кімнатний пристрій, для якого характерна компактність і доступна вартість. Має локальне значення, тобто визначає параметри конкретної місцевості. Основне завдання полягає у вимірі температурної позначки. До плюсів можна віднести малі розміри та вагу, до недоліків – невисокий рівень точності.

3. Гігрометр. Це додаткова функція виробів, що дозволяє їм вимірювати рівень вологості у приміщенні та покращувати стан здоров'я. Показники можуть вимірюватися та аналізуватися у комплексі, що сприяє формуванню комфортного мікроклімату згодом.

4. Датчик. Це термометри, оснащені додатковим приладом, що прикріплюється до зовнішньої частини віконної конструкції. Мета – вимірювання показань за вікном. Є системи провідного та бездротового типу. Такі моделі визначають ступінь комфортності мікроклімату, виносять попередження та повідомляють про зміни.

5. Прогноз погоди. Таку опцію мають цифрові пристрої. Вони збирають інформацію про основні параметри клімату та на її підставі планують подальший розвиток подій. Важливу роль відіграє раціональна установка сенсорів, впоратися із цим завданням може виключно фахівець.

6. Символьне відображення. Воно може бути цифровим та символьним. Відомості про погоду з'являються на екранах і є певними символами. Об'єкти змінюються відповідно до змін клімату.

7. Годинник. Такий пристрій є багатофункціональним та вирішує практично всі завдання, зазначені вище. Воно може забезпечуватися класичними та проекційними хронометрами. Відбувається автоматична синхронізація із радіосигналами.

Як можна помітити, види метеостанцій численні, але це не всі рішення, оскільки деякі пристрої можуть містити набір додаткових функцій. Зокрема, йдеться про такі можливості, як контроль погоди, календар місячного та сонячного типу, використання пульта, синхронізація з комп'ютером, заряд батареї за допомогою сонця, компас, індикація екрану та ін.

***Станція М-107***

Основні технічні характеристики

1. Атм. тиск 600-1050 гПа (1.0 гПа)
2. Температура повітря $50^0$C ($80^o$C)
3. Швидкість вітру за 10 хв. 1-40 м/сек (1,0 м-сек)

4. Напрям вітру 0-360$^0$ ( 100)

5. Кількість рідких опадів за 3 чи 6 год до 50мм (0,5мм)

6. Сонячне сяйво (наявність/відсутність)

7. Туман у межах 200-600 м (наявність/відсутність)

Для роботи в необжитих районах має специфічне джерело живлення (вітроелектричні агрегати, ізотопні джерела, пристрої, що використовують газ пропан) та буферну акумуляторну батарею.

Робота станції М-107 за 1 цикл.

Між термінами, до передачі виміряних значень метеоелементів, апаратура станції забезпечує підрахунок кількості рідких опадів, що випали, за 3 години. Робочий цикл станції починається за 12 хв. До початку передачі інформації відбувається в наступній послідовності.

1. Контактами автоспуску подається напруга на анемометр протягом 10 хв.

2. З 30 с до включення станції відбувається підзавод годинникового механізму. І подається перший пусковий імпульс, при якому вмикається 1-е головне реле блоку автоматики. При спрацьовуванні головного реле включається стабілізатор блоку живлення, блок вимірювання температури повітря, датчик атмосферного тиску, датчик туману, сонячного сяйва та вимикається живлення датчика опадів.

3. Після закінчення 30 с від автопуску відбувається другий пусковий імпульс, що надходить в блок автоматики. Подається живлення на блок управління та кодування, одночасно живлення надходить у блок радіопередавачів.

4. Після передачі кожен лічильник встановлюється у вихідне положення. Знімається блокування та відключаються всі ланцюги блоку автоматики, відключаються ланцюги живлення радіопередавачів. Включається ланцюг вимірювання опадів. Схема повертається у вихідне становище.

***Метеостанція WH-5302-1***

Включає Рис. 15.1):

- 2 сенсора температури від -40 до +65$^o$C
- сенсор відносної вологості від 1 до 99%
- сенсор швидкості повітря
- сенсор опадів.

Бездротовий спосіб передачі значень погодних параметрів з сенсорів на комп’ютер.


Рисунок 15.1 Метеостанція WH-5302-1

***Автономна професійна метеостанція з бездротовою передачею інформації з датчиків, сенсорним керуванням базою та USB-інтерфейсом SITITEK AW002.***

SITITEK AW002 — відмінно укомплектована автономна (живлення датчиків від сонячної батареї, через акумуляторні батареї) метеостанція з сенсорним дисплеєм, USB-інтерфейсом (програмне забезпечення в комплекті), з бездротовим способом (до 100 метрів) передачі значень погодних параметрів радіозв'язку (Рис. 15.2).

Програмне забезпечення Easyweather через USB-інтерфейс дозволяє зареєструвати метео-точку та передавати дані на погодні хмарні сервіси в режимі реального часу, наприклад, wunderground.com та інші. Надалі доступ до баз даних можна отримати через смартфони, ПК і т.д.

Професійна метеостанція SITITEK AW002 комплектується всіма необхідними для аналізу погоди датчиками (швидкість та напрям вітру, температура, вологість, кількість опадів, тиск). Крім виміру погодних умов прилад має функції оповіщення при зміні атмосферного тиску, напрямку вітру та інших показників, які можна самостійно налаштувати. SITITEK AW002 автоматично аналізує та прогнозує зміни погоди.

Рисунок 15.2 Метеостанція SITITEK AW002

Технічні характеристики:
• вимірювання температури повітря на вулиці: від -40 до +65 °C
• вимірювання температури повітря у приміщенні: від 0 до +50 °C
• одиниці вимірювання температури: °C або °F
• вимір відносної вологості повітря: від 10 до 99%
• точність вимірювання вологості: 1%
• вимір опадів: від 0 до 9999 мм
• визначення напрямку та швидкості вітру
• вимір атмосферного тиску
• індикація мінімальних та максимальних значень
• прогноз погоди
• календар
• годинник
• дисплей бази метеостанції: РК, сенсорний
• робоча частота передачі: 433 МГц
• радіус дії бездротової передачі даних: до 100 метрів
• індикація мінімальних та максимальних значень із фіксацією часу
• сонячна зарядка акумулятора для бездротової передачі даних

У комплекті поставки: програмне забезпечення, комплект датчиків, акумулятори, база із сенсорним дисплеєм, інструкція по встановленню, комплект батарейок 3+2 (акумуляторні).

У теперішній час у зв’язку з розвитком **вільного API** бурно розвиваються власні розробки апаратури на мікропроцесорній техніці, ціна якої значно менше за комерційну, зокрема і метеостанцій. API -- (інтерфейс прикладного програмування) спрощує процес програмування під час створення програм, абстрагуючи базову реалізацію і надаючи лише об'єкти чи дії, необхідні розробнику. Якщо графічний інтерфейс для поштового клієнта може надати користувачеві кнопку, яка виконає всі кроки для вибірки та виділення нових листів, то API для вводу/виведення файлів може дати розробнику функцію, яка копіює файл з одного місця в інше, не вимагаючи від розробника розуміння операційної файлової системи, що відбуваються за лаштунками.

Ідея вільної та доступної інформації про погоду призвела до того, що створили та надали всім розробникам програм безкоштовний API для отримання різноманітних даних про погоду, такими як (Рис. 15.3):

— Інтерактивна карта з даними про поточну погоду

— Прогноз на тиждень у місті

— Історичні дані у 120 000 містах світу.

— Дані від 40 000 метеостанцій по всьому світу, що отримуються практично в режимі online. (Затримка від секунд до години)

- Багато різних web карт, включаючи карти хмар, опадів, вітру, температури тощо.

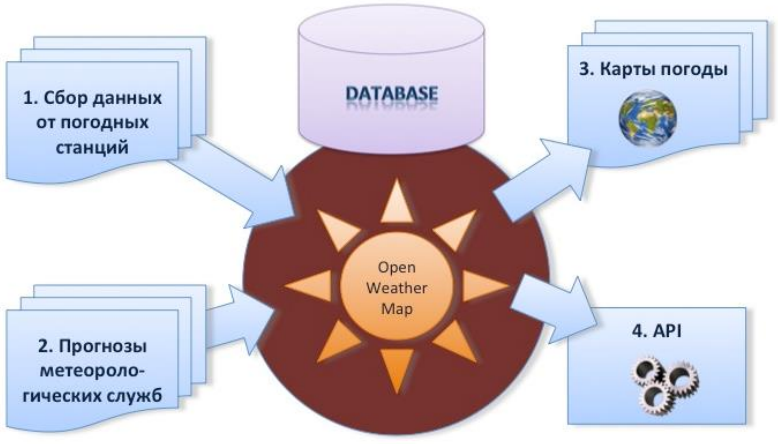



Рисунок 15.3 Ідея вільної та доступної інформації про погоду

Аматорські погодні станції це взагалі окрема тема. Але в цій статті хотілося б наголосити, що спектр таких станцій дуже широкий. І цікаво буде встановити таку станцію у себе вдома чи на дачі не лише серйозному

радіоаматору, а й, наприклад, татові із сином. Можна купити готову станцію вартістю від $100 до $1000, або зібрати самому, наприклад, Arduino (Рис. 15.4).

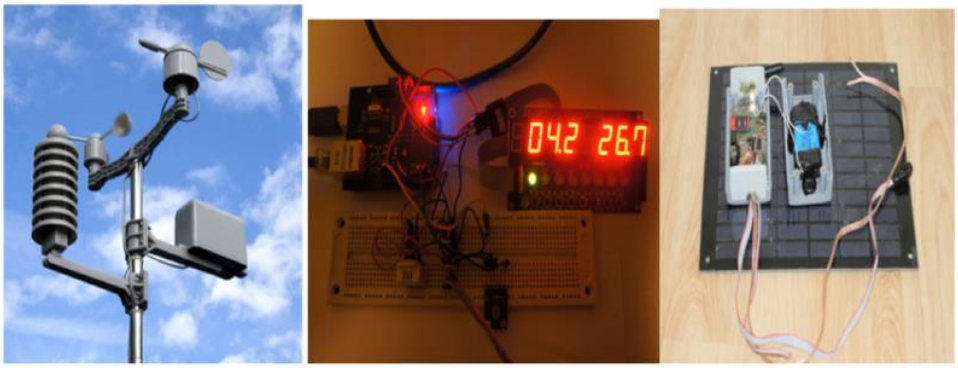



Рисунок 15.4 Метеостанція на Arduino

Спектр застосування API нескінченно широкий. Це мобільні програми для всіх платформ. Це різноманітні web-сайти, які можуть використовувати API для відображення поточної погоди, різних погодних графіків, віджетів тощо. Це системи розумного будинку.

Наприклад, один із користувачів OpenWeatherMap з Великобританії організував систему автоматичного поливу свого англійського садка (Рис. 15.5). Для планування кількості води та режиму поливу він використовує дані про прогноз опадів.

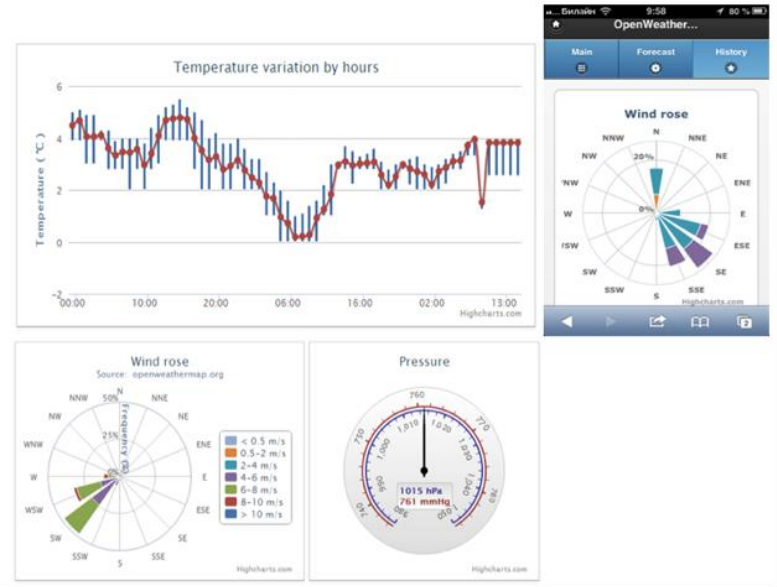



Рисунок 15.5 Інтерфейс метеостанції

Сама ідея вільної погоди дає змогу розвитку проекту у нових напрямах і функціях.

# ВИКОРИСТАНІ ДЖЕРЕЛА

## Основна


1. Плескач В.Л. Інформаційні системи і технології на підприємствах: підручник /В.Л. Плескач, Т.Г. Затонацька. К.: Знання, 2011. 718с.
2. Основи інформаційних технологій і систем: підручник / В. А. Павлиш, Л. К. Гліненко, Н. Б. Шаховська. Львів: Львівська політехніка, 2018. 620 с.
3. Основи інформаційних технологій і систем: навч. посіб. / В. А. Павлиш, Л. К. Гліненко ; М-во освіти і науки України, Нац. ун-т «Львів. політехніка». Л. : Вид-во Львів. політехніки, 2013. 500 с.
4. Информационные системы / Петров В. Н. СПб.: Питер, 2003. 688 с.
5. Павловец П. В. Микропроцессорные средства и системы : учеб.-метод. комплекс для студентов специальности «Вычислительные машины, системы и сети». – Новополоцк: ПГУ, 2011. – 364 с.
6. Карпов В.Е., Коньков К.А. Основы операционных систем. Курс лекций. М.: ИНТУИТ.РУ «Интернет Университет Информационных Технологий», 2005. 536 с.
7. Романова Ю.Д. Моя первая книга об информационных технологиях. М.: Эксмо. 2006. 400 с.
8. Советов Б. Я., Яковлев С. А. С 56 Моделирование систем: Учеб. для вузов 3-е юд., перераб. и доп. М.: Высш. шк., 2001. 343 с.
9. Таненбаум Э. Архитектура компьютера СПб.: Питер, 2003. 695 с.
10. Информатика. Базовый курс. 2-е издание / Под ред. С. В. Симоновича. СПб.: Питер, 2005. 640 с.


## Додаткова


11. Мелехин В.Ф., Павловский Е.Г. Вычислительные машины, системы и сети. Учебник. М.: Издательский центр “Академия”, 2006. – 560с.
12. Глушаков С.В., Мельников И.В., Персональный компьютер: Учебный курс. - Харьков: Фолио; М.: ООО «Изд-во АСТ»,2001.- 520 с.
13. Догадин Н.Б. Архитектура компьютера / Догадин Н.Б., - 3-е изд., (эл.) - М.:БИНОМ. ЛЗ, 2015. - 274 с.: ISBN 978-5-9963-2638-9 URL:http://znanium.com/bookread2.php?book=539585.
14. Максимов Н. В. Архитектура ЭВМ и вычислительных систем: Учебник /Н.В. Максимов, Т.Л. Партыка, И.И. Попов. - М.: Форум:НИЦ ИНФРА-М, 2013. 512 с. URL:http://www.znanium.com/bookread.php?book=405818

Навчальне електронне видання

МЕЩЕРЯКОВ Володимир Іванович
ЛАШИНА Катерина Валерієва

# ІНФОРМАЦІЙНІ СИСТЕМИ І ТЕХНОЛОГІЇ

Конспект лекцій

**Видавець і виготовлювач**
Одеський державний екологічний університет
вул. Львівська, 15, м. Одеса, 65016
тел./факс: (0482) 32-67-35
E-mail: info@odeku.edu.ua
Свідоцтво суб'єкта видавничої справи
ДК № 5242 від 08.11.2016