МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

Т. А. Григорова

АРХІТЕКТУРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

Навчальний посібник (електронне видання)

Кременчук
2024

УДК 378.14.015.62

Рекомендовано до друку вченою радою
Кременчуцького національного університету імені Михайла Остроградського
(Протокол № 4 від 24.12. 2024 року)

Рецензенти:

Гвоздева Ірина Маратовна, доктор технічних наук, професор, завідувач кафедри електрообладнання і автоматики суден

Чорна Ольга Анатоліївна, кандидат технічних наук, доцент, доцент кафедри комп'ютерної інженерії та електроніки

Григорова Т. А.
Архітектура комп'ютерних систем та мереж: навчальний посібник (електронне видання). Кременчук: б.в., 2024. с.

Посібник спрямований на формування у здобувачів освіти системи знань з інформатики, практичних умінь у здійсненні навчальної діяльності у закладах загальної середньої, фахової передвищої освіти, формування професійної компетентності майбутнього вчителя інформатики й готовності до використання інформаційних технологій навчання у майбутній професійній діяльності. До розділів посібника підібрані завдання, виконання яких сприятиму більш глибокому розумінню теоретичного матеріалу, а контрольні питання сприятиму перевірки засвоєних знань. Все це дозволить в умовах сучасного реформування освітньої діяльності, здобути знання і виробити вміння і навички для успішної самореалізації в майбутній професійній діяльності.

Для студентів денної форми навчання зі спеціальності 014.09 «Середня освіта (Інформатика)» освітньо-професійної програми «Середня освіта (Інформатика) першого (бакалаврського) освітнього рівня.

УДК 378.14.015.62

ISBN

© Т. А. Григорова, 2024
© Кременчуцький національний університет імені Михайла Остроградського, 2024

ЗМІСТ

ВСТУП 4

РОЗДІЛ 1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО ЕОТ 5
1.1. Класифікація ЕОМ 5
1.2. Загальні принципи побудови сучасних ЕОМ 8
ЗАВДАННЯ 19
КОНТРОЛЬНІ ПИТАННЯ 19

РОЗДІЛ 2. ОРГАНІЗАЦІЯ КОМП'ЮТЕРНИХ СИСТЕМ, ЯК КОМПЛЕКСУ ТЕХНІЧНИХ ЗАСОБІВ 19
2.1. Принципи побудови комп'ютерів. 19
2.2. Паралелізм на рівні процесорів 21
2.3. Побудова процесора 22
2.4. Взаємодія елементів при роботі мікропроцесора 23
2.5. Склад та принцип дії основної пам'яті 24
2.6. Розміщення інформації в основний пам'яті ЕОМ 27
2.7. Принципи організації підсистеми введення-виведення 28
2.8. Багаторівнева структура ЕОМ 40
2.9. Цифровий логічний рівень 41
2.10. Мікроархітектурний рівень 44
2.11. Рівень архітектури команд 45
ЗАВДАННЯ 53
КОНТРОЛЬНІ ПИТАННЯ 53

РОЗДІЛ 3. РІВЕНЬ ОПЕРАЦІЙНОЇ СИСТЕМИ 53
3.1. Режими роботи ЕОМ 53
3.2. Структура ОС Windows 58
3.3. Історія створення ОС Windows 58
3.4. Особливості ОС Windows 59
3.5. Архітектура ОС Windows 61
3.6. Підсистема Win32 і Win64 63
ЗАВДАННЯ 64
КОНТРОЛЬНІ ПИТАННЯ 64

РОЗДІЛ 4. МЕРЕЖЕВИЙ РІВЕНЬ 64
4.1. Класифікація комп'ютерних мереж 65
4.2. Стандартизація у комп'ютерних мережах 69
4.3. Семирівнева модель OSI (Open Systems Interconnection) 70
4.4. Протоколи і стеки протоколів. 74
ЗАВДАННЯ 75
КОНТРОЛЬНІ ПИТАННЯ 76

СПИСОК ЛІТЕРАТУРИ 77

ВСТУП

Навчальний посібник «Архітектура комп'ютерних систем та мереж» направлений на те, щоб дати здобувачам освіти основи знань з інформатики, а саме архітектури комп'ютера, архітектури операційної системи, архітектури комп'ютерної мережі. Інформація представлена в підручнику допоможе студенту отримати знання про роботу комп'ютера, взаємодію його пристроїв і обчислювальних систем в процесі обробки інформації на всіх рівнях комп'ютерної архітектури: цифровому, логічному, рівні мікрокоманд, системи команд, рівні архітектурної підтримки механізмів операційних систем та комп'ютерних мереж. Вона сприятиме формуванню професійної компетентності майбутнього вчителя інформатики.

Посібник має чотири розділи, які відображають багаторівневу структуру комп'ютера. В кінці кожного розділу сформовані практичні завдання і контрольні питання, які допоможуть студенту краще засвоїти теоретичний матеріал.

У першому розділі іде мова про загальну інформацію о комп'ютерах їх класифікацію за різними підходами, які існують на сьогоднішній день. Крім того значна увага приділяється загальним принципам побудови ЕОМ з точки зору обробки інформації, яким чином обробляється інформація, в якому вигляді, що таке машинний код і як він записується.

У другому розділі основна увага приділяється технічним засобам з яких складається комп'ютер, таким як процесор, оперативна пам'ять, підсистемі введення-виведення, інтерфейсам взаємодії між всіма пристроями комп'ютера, міжкомп'ютерним інтерфейсам, а також інтерфейсам взаємодії комп'ютера з користувачем. Розглядається питання багаторівневої структури ЕОМ, увага приділяється трьом нижнім рівням які характеризують безпосередньо архітектуру комп'ютера. В підрозділі «Рівень архітектури команд» розглядається мова асемблера, що призначена для написання програм, які безпосередньо керують апаратним забезпеченням.

Третій розділ містить інформацію про рівень операційної системи на прикладі операційної системи Windows, її структури, історії її створення і її архітектури.

Останній четвертий розділ містить інформацію про мережевий рівень, який є невід'ємною частиною сучасних ЕОМ.

РОЗДІЛ 1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО ЕОТ

1.1. Класифікація ЕОМ

Класифікація комп'ютерів це поділ комп'ютерних систем на різні категорії за певними критеріями. Така класифікація дозволяє зрозуміти, для яких цілей використовується та чи інша система, а також її продуктивність, розмір, тип задач та інші характеристики.

Щороку вартість обчислень скорочується приблизно на 25-30%, вартість зберігання одиниці інформації – до 40%. Постійно змінюється покоління машин, які будуються на основі нових типів мікропроцесорів, що визначають характеристики ЕОМ.

У цих умовах будь-яка запропонована класифікація ЕОМ дуже швидко застаріває і потребує корегування. Наприклад, в класифікаціях десятирічної давності широко використовувалися назви міні-, міді- і мікроЕОМ, які майже зникли з ужитку. Разом з тим існує цілий ряд закономірностей обчислювальної техніки, які дозволяють передбачити основні тенденції розвитку ЕОМ. Необхідно аналізувати традиційні і нові області застосування ЕОМ, класи і типи використовуваних обчислювальних засобів, поточну кон'юнктуру ринку інформаційних технологій і його динаміку.

Академік Віктор Михайлович Глушков, видатний вчений у галузі кібернетики та комп'ютерних наук, засновник Інституту кібернетики імені В. М. Глушкова НАН України, запропонував поділ використання різних типів електронних обчислювальних машин (ЕОМ) на три основні глобальні сфери діяльності людини. Цей поділ дає можливість систематизувати та визначити напрямки застосування комп'ютерних технологій у різних галузях.

– Автоматизація праці та виробництва. Використання ЕОМ для автоматизації процесів на виробництві, управлінні, проектуванні та плануванні. Це включає застосування комп'ютерних систем для автоматизованого керування, аналізу даних і оптимізації технологічних процесів, що дозволяє підвищити ефективність і зменшити людську помилку.

– Обробка і зберігання інформації. Комп'ютери використовуються для зберігання, обробки, аналізу та пошуку інформації. Це включає створення інформаційних систем, баз даних, архівів, а також інтелектуальних систем, що допомагають у прийнятті рішень на основі великої кількості даних.

– Моделювання та наукові дослідження. ЕОМ активно застосовуються в науці та дослідженнях для моделювання складних процесів, що неможливо або важко змоделювати в реальному світі. Це включає чисельні методи для фізичних, хімічних, біологічних досліджень, а також розробку нових теоретичних моделей у різних наукових дисциплінах. Особливістю цього напрямку є наявність математичної основи, яка розвивалась з розвитком математичних наук і їх додатків.

Глушков передбачав, що в майбутньому комп'ютери будуть використовуватися в усіх аспектах людської діяльності, від повсякденних завдань до складних наукових досліджень, що й підтвердило розвиток комп'ютерних технологій і обчислювальних систем.

Перший і другий напрямки застосування ЕОМ пов'язані з використанням систем управління. Він народився приблизно в 60-і роки, коли ЕОМ стали інтенсивно впроваджуватися в контури керування автоматичних і автоматизованих систем. Це сприяло розвитку ЕОМ в напряму багато програмних, багатокористувацьких і сполучених каналами зв'язку комп'ютерних систем. Змінилась структуру комп'ютерів за рахунок появи блоків переривань і пріоритетів, блоків захисту і т.п. З'явилась різноманітна периферія і спеціальні процесори або канали введення-виведення даних.

Нової сфері робіт в найбільшій мірі відповідали міні-ЕОМ (термін цей з'явився в 1968 році). Саме вони стали використовуватися для управління галузями, підприємствами, корпораціями. Машини нового типу задовольняли наступним вимогам:

– більш дешеві в порівнянні з великими ЕОМ, що забезпечували централізовану обробку даних;

– більш надійні, особливо при роботі в контурі управління;

– мали гнучкі і адаптовані настройки на конкретні умови функціонування;

– мали архітектурну прозорість, тобто структура та функції ЕОМ були зрозумілі користувачам.

На зміну міні-ЕОМ прийшли ЕОМ інших типів: сервери, середні ЕОМ або старші моделі персональних ЕОМ (ПЕОМ).

Одночасно зі структурними змінами ЕОМ відбувалося і якісна зміна характеру обчислень. Частка чисто математичних розрахунків постійно скорочувалася, і в даний час вона становить близько 10% від всіх обчислювальних робіт. Машини все більше стали використовуватися для нових видів обробки: текстів, графіки, звуку та ін.

З'явився ще один напрямок пов'язаний із застосуванням ЕОМ для вирішення завдань штучного інтелекту. Завдання штучного інтелекту припускають отримання не точного результату, а найчастіше усередненого в статистичному, імовірнісному сенсі. До цих завдань відносяться: завдання робототехніки, доведення теорем, машинного перекладу текстів з однієї мови на іншу, планування з урахуванням неповної інформації, складання прогнозів, моделювання складних процесів і явищ і т.д. Цей напрямок все більше набирає силу. У багатьох областях науки і техніки створюються і удосконалюються бази даних і бази знань, експертні системи. Для технічного забезпечення цього напрямку використовуються якісно нові структури ЕОМ з великою кількістю обчислювачів (ЕОМ або процесорних елементів), що забезпечують паралелізм в обчисленнях.

Цей невеликий перелік областей застосування ЕОМ показує, що для вирішення різних завдань потрібна різна обчислювальна техніка. Комп'ютерів сьогодні розрізняються за призначенням, потужністю, розмірами, елементній базі і т.д. тому класифікують ЕОМ за різними ознаками. Будь-яка класифікація є певною мірою умовна, оскільки розвиток комп'ютерної науки і техніки дуже стрімкий. Крім того, зарахування комп'ютерів до певного класу досить умовне як через нечіткість розмежування груп, так і в наслідок впровадження в практику замовного складання комп'ютерів, де номенклатуру вузлів і конкретні моделі адаптують до вимог замовника. Розглянемо найбільш поширені критерії класифікації комп'ютерів.

Класифікація за призначенням

– Суперкомп'ютери, що призначенні для виконання надскладних обчислень, моделювання природних явищ, прогнозування погоди, ядерні та космічні дослідження, вони мають тисячі або мільйони процесорів, здатні виконувати трильйони операцій за секунду (FLOPS) (IBM Summit, Fugaku (Японія));

– Мейнфрейми (Mainframe) призначенні для обробка великої кількості транзакцій, підтримка багатокористувацьких систем (банки, фінансові установи, державні організації), є надійні, безпечні та здатні працювати без збоїв роками (IBM Z-серія);

– Міні-комп'ютери (Minicomputer) використовуються на виробництвах, заводах для автоматизації процесів, управління обладнанням, є більш потужні за ПК, але менш продуктивні за мейнфрейми (DEC PDP-11, VAX-системи);

– Персональні комп'ютери (ПК) використовуються звичайними користувачами для офісної роботи, навчання, розваг, програмування тощо, є універсальні, доступні за ціною, мають широкий спектр використання (стаціонарні комп'ютери (desktop), ноутбуки);

– Мобільні комп'ютери, їх основна характеристика мобільність і портативність для користувачів, вони компактні, енергоефективні, можуть працювати без підключення до електромережі (лептопи, планшети, смартфони, смарт-годинники).

Мейнфрейми (Mainframe) застосовують для обслуговування великих галузей народного господарства. Вони характеризуються 64-розрядними паралельно працюючими процесорами (кількість яких досягає до 100), інтегральною швидкодією до десятків мільярдів

операцій в секунду, розрахованих на багатокористувацький режим роботи. Домінуюче роль у випуску комп'ютерів такого класу займає фірма IBM (США).

Використовуються для створення обчислювальних центрів, які містить кілька відділів або груп. Штат обслуговування – десятки людей.

Група технічного обслуговування – Центральний процесор – Група підготовки даних

Група системних програмістів – Відділ видачі результатів – Група прикладних програмістів

Група інформаційної підтримки

Центральний процесор – основний блок ЕОМ, в якому відбувається обробка даних і обчислення результатів. Являє собою декілька системних блоків в окремій кімнаті, де підтримується постійна температура і вологість повітря.

Група системного програмування – займається розробкою, налагодженням і впровадженням програмного забезпечення, необхідного для функціонування обчислювальної системи. Системні програми забезпечують взаємодію програм з обладнанням, тобто програмно-апаратний інтерфейс обчислювальної системи.

Група прикладного програмування – займається створенням програм для виконання конкретних дій з даними, тобто забезпечення користувацького інтерфейсу обчислювальної системи.

Група підготовки даних – займається підготовкою даних, які будуть опрацьовані на прикладних програмах, створених прикладними програмістами. Зокрема, це набір тексту, сканування зображень, заповнення баз даних.

Група технічного забезпечення – займається технічним обслуговуванням всієї обчислювальної системи, ремонтом та налагодженням апаратури, під'єднанням нових пристроїв.

Група інформаційного забезпечення – забезпечує технічною інформацією всі підрозділи обчислювального центру, створює і зберігає архіви розроблених програм (бібліотеки програм) та накопичених даних (банки даних).

Відділ видачі даних – отримує дані від центрального процесора і перетворює їх в форму, зручну для замовника (роздрук).

Великим ЕОМ притаманна висока вартість обладнання та обслуговування, тому робота організована у неперервний цикл.

Персональні комп'ютери умовно можна поділити на професійні та побутові, але в зв'язку з здешевленням апаратної частини, межі між ними розмиваються. З 1999 року задіяний міжнародний сертифікаційний стандарт - специфікація PC99:

- масовий персональний комп'ютер (Consumer PC)
- діловий персональний комп'ютер (Office PC)
- портативний персональний комп'ютер (Mobile PC)
- робоча станція (WorkStation)
- розважальний персональний комп'ютер (Entertaiment PC)

Більшість персональних комп'ютерів на ринку підпадають до категорії масових ПК. Ділові ПК - мають мінімум засобів відтворення графіки та звуку. Портативні ПК відрізняються наявністю засобів з'єднання віддаленого доступу (комп'ютерний зв'язок). Робочі станції - збільшені вимоги до пристроїв зберігання даних. Розважальні ПК - основний акцент до засобів відтворення графіки та звуку.

Класифікація за архітектурою

– Цифрові комп'ютери обробляють інформацію у вигляді дискретних чисел (двійковий код), використовуються у більшості сучасних комп'ютерів (ПК, ноутбуки, смартфони);

– Аналогові комп'ютери обробляють аналогові (безперервні) сигнали, що часто використовується в наукових дослідженнях та вимірювальних приладах, Використовуються для роботи з фізичними величинами (температура, тиск, звук);

– Гібридні комп’ютери поєднують властивості цифрових та аналогових комп’ютерів, використовуються у медичній техніці (КТ, МРТ) та спеціалізованих системах керування.

Класифікація за продуктивністю

– Суперкомп’ютери – найвища продуктивність (від кількох петафлопс до ексафлопс);

– Мейнфрейми – висока продуктивність, здатні обробляти сотні тисяч транзакцій одночасно;

– Сервери – висока продуктивність, здатні працювати 24/7 у режимі великого навантаження;

– Персональні комп’ютери (ПК) – помірна продуктивність для щоденного використання;

– Мобільні пристрої – смартфони, планшети, з обмеженою продуктивністю через малий розмір та енергоспоживання.

Класифікація за розміром

– Мікрокомп’ютери – найменші пристрої (Arduino, Raspberry Pi), використовуються для автоматизації та IoT;

– Персональні комп'ютери – класичні ПК та ноутбуки для індивідуальних користувачів;

– Міні-комп’ютери – більші за ПК, використовуються для управління промисловими процессами;

– Мейнфрейми – великі сервери, що займають цілі кімнати;

– Суперкомп’ютери – великі обчислювальні комплекси, що займають кілька залів.

Класифікація за способом керування

– Керовані користувачем – користувачі самі виконують управління пристроєм;

– Автоматизовані – програми автоматично виконують завдання без постійного втручання користувача (роботизовані системи, автоматизовані лінії виробництва).

Класифікація за типом використання

– Загального призначення – використовуються для широкого кола завдань (ПК, ноутбуки, смартфони);

– Спеціального призначення – використовуються для специфічних завдань (контролери на заводах, медичне обладнання (КТ, МРТ).

Класифікація за рівнем інтелектуальності

– Традиційні комп'ютери – виконують програми, але не здатні "мислити" самостійно;

– Штучний інтелект (AI-системи) – здатні до самонавчання та прийняття рішень;

Класифікація за способом взаємодії

– Локальні комп'ютери – використовуються локально, без підключення до мережі;

– Мережеві комп'ютери – підключені до локальної мережі або інтернету для спільного використання ресурсів;

– Термінальні комп'ютери – підключаються до основного сервера для обробки даних.

1.2. Загальні принципи побудови сучасних ЕОМ

Основним принципом побудови всіх сучасних ЕОМ є програмне керування. В основі його лежить представлення алгоритму розв’язання будь-якої задачі у вигляді програми обчислень.

Згідно стандарту (ISO 2382/1-84) *алгоритм* – кінцевий набір розпоряджень, що визначає рішення задачі за допомогою кінцевою кількості операцій, а *програма* (для ЕОМ) – впорядкована послідовність команд, що підлягає обробці. Слід зауважити, що суворого, однозначного визначення алгоритму, так само як і однозначних методів його перетворення, в програму обчислень, не існує. Принцип програмного управління може бути здійснено різними способами. Стандартом для побудови практично всіх ЕОМ став принцип, описаний Дж. Фон Нейманом в 1945 р, який використовувався при побудові ще перших зразків ЕОМ (рис. 1.1).

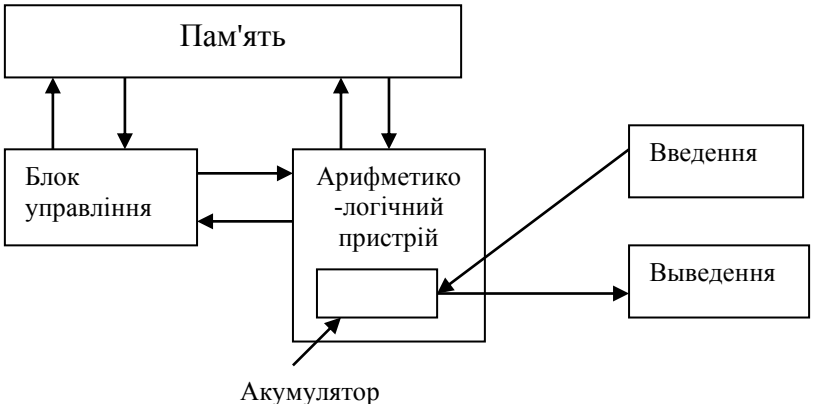

Рисунок 1.1 Схема Дж. Фон Неймана

Пам’ять включала 4096 слів, розмір слова складався з 40 бітів, що налічувало 2 команди по 20 бітів або ціле число. Кожна команда складалась з типу команди розміром 8 бітів, 12 бітів визначало слово, яке означало чим оперує команда. Акумулятор це особливий внутрішній регістр в 40 бітів в який типова команда додавала слово з пам’яті або зберігала його вміст в пам’яті.

Всі обчислення, визначені алгоритмом рішення завдання, представлені у вигляді програми, що складається з послідовності керуючих слів-команд. Кожна команда містить вказівку на виконувану *операцію* та місце знаходження операндів і ряд службових ознак. *Операнди* – змінні, значення яких використовуються в операціях перетворення даних. Доступ до команд і операндів, з яких складається програма, відбувається за вказівкою на їх *адреси* – номери комірок пам’яті ЕОМ, призначених для зберігання об’єктів.

Інформація (команди і дані: числові, текстові, графічні тощо) кодуються двійковими цифрами 0 і 1, тобто представлені у двійковій системі числення. Тому різні типи інформації, розміщені в пам’яті ЕОМ, практично неможливо розрізнити, ідентифікація їх можлива лише при виконанні програми, згідно з її логікою, по контексту. Кожен тип інформації має свій формат – структурну одиниці інформації. Зазвичай всі формати даних, що використовуються в ЕОМ, кратні байту (8 бітів), тобто складаються з цілого числа байтів.

Послідовність бітів у форматі, що має певний сенс, називається *полем*. Наприклад, в кожній команді програми розрізняють поле коду операцій, поле адреси операндів. В числовій інформації виділяють знакові розряди, поле значущих розрядів чисел, старші та молодші розряди.

Послідовність числа байтів, яку може обробити ЕОМ за одну операцію називається *словом*. В якості структурних елементів інформації розрізняють також півслова, подвійне слово і ін.

Окрім двійкової системи числення – {0,1} використовується шістнадцяткова – {0,1,...,9, A, B, C, D, E, F} де цифра A позначає число 10, F – число 15 і вісімкова – {0, 1, 2, 3, 4, 5, 6, 7}. Дві останні використовуються для опису адреси команди і для компактнішого зображення двійкової інформації, оскільки запис значення чисел здійснюється суттєво меншим числом знаків.

Вісімкова і шістнадцяткова системи числення є похідними від двійкової, оскільки $16 = 2^4$ і $8 = 2^3$.

Приклад 1.1. Число 100,625 у двійковій, вісімковій і шістнадцятковій системах числення має таке відображення:

$A_2 = 1100100{,}101$; $A_2 = 1*2^6+1*2^5+0*2^4+0*2^3+1*2^2+0*2^1+0*2^0+1*2^{-1}+0*2^{-2}+1*2^{-3}$;

$A_8 = 144.5$; $A_8 = 1*8^2+4*8^1+4*8^0+5*8^{-1}$;

$A_{16} = 64.A$; $A_{16} = 6*16^1+4*16^0+10*16^{-1}$.

Відображення чисел у різних системах числення допускає однозначне перетворення їх з однієї системи на іншу.

Для перетворення шістнадцяткової (вісімкової) системи числення на двійкову достатньо двійковим кодом записати шістнадцяткові коди цифр тетродами (по 4 двійкові розряди) і тріадами (по 3 двійкові розряди) – для вісімкових цифр. Обернене перетворення з двійкового коду проводять у зворотному порядку: двійкове число розділяють вліво і вправо від межі цілої та дробової частин на тетроди – для подальшого запису чисел у шістнадцятковому зображенні, на тріади – для запису їх значень вісімковими цифрами.

Ціле число з основою N_1 переводять в систему числення з основою N_2 завдяки послідовному діленню числа A_n, на основу N_2, записану у вигляді числа з основою N_1, до отримання остачі. Цей процес повторюють до тих пір, поки частка не стане меншою за дільник. Отримані остачі від ділення та останню частку записують в оберненому порядку

Приклад 1.2 $A_{10} = 37$ $A_2 = ?$ $A_{16} = ?$

1. 37/2
   1 18/2
   0 9/2
   1 4/2
   0 2/2
   0 1

$A_2 = 100101$

2) 37/16
   5 2

$A_{16} = 25$

Дробове число з основою N_1 переводиться в систему числення з основою N_2 завдяки послідовному множенню A_n на основу N_2, записану у вигляді числа з основою N_1. Цілу частину добутку беруть у вигляді наступної цифри відповідного розряду, а дробову частину, що залишилася, беруть як новий множник.

Приклад 1.3 $A_{10} = 0.625$ $A_2 = ?$ $A_8 = ?$ $A_{16} = ?$

	0,625		0,625		0,625
	x 2		x 8		x 16
1	250	5	000	10	000
	x 2	$A_8 = 0,5$		$A_{16} = 0,A$	
0	500				
	x 2				
1	000				

$A_2 = 0,101$

Усі ЕОМ (електронні обчислювальні машини) без винятку виконують операції з числами у вигляді спеціальних машинних кодів. Їх використання дозволяє обробляти знакові розряди чисел так само, як і значущі розряди, а також замінювати операцію віднімання операцією додавання.

Розрізняють прямий код (П), обернений код (ОК) і додатковий код (ДК) двійкових чисел.

Прямий код двійкового числа утворюють з абсолютного значення цього числа та коду знака (нуль або одиниця) перед його старшим числовим розрядом.

Приклад 1.4

$A10 = +10$ $A2 = +1010$ $[A2]_П = 0:1010;$

$B_{10} = -15$ $B_2 = -1111$ $[B_2]_П = 1:1111.$

Обернений код двійкового числа утворюють за таким правилом. Зворотний код додатних чисел збігається з їх прямим кодом. Обернений код від'ємного числа містить одиницю в знаковому розряді числа, а значущі розряди числа замінюють на інверсні, тобто нулі замінюють одиницями, а одиниці – нулями.

Приклад 1.5

$A_{10} = +5$ $A_2 = +101$ $[A_2]_П = [A_2]_{ОК} = 0:101;$

$B_{10} = -13$ $B_2 = -1101$ $[B_2]_{ОК} = 1:0010.$

Основні властивості оберненого коду:

– додавання додатного числа з його від'ємним значенням в оберненому коді дає, так звану машинну одиницю МОок = 1: 111... 11, що складається з одиниць у знаковому і значущих розрядах числа;

– нуль в оберненому коді має двояке значення. Він може бути додатним – 0: 00...0 та від'ємним числом – 1; 11... 11. Значення від'ємного нуля збігається з МОок. Подвійне зображення нуля виявилося причиною того, що всі числа зображують не оберненим, а додатковим кодом.

Додатковий код додатних чисел збігається з їх прямим кодом. Додатковим кодом від'ємного числа є результат додавання оберненого коду числа з одиницею молодшого розряду (2° – для цілих чисел, 2-k – для дробових).

Приклад 1.6.

$A_{10} = +19$ $A_2 = +10011$ $[A_2]_П = [A_2]ОК = [A_2]_{ДК} = 0{:}10011;$

$B_{10} = -13$ $B_2 = -1101$ $[B_2]_{ДК} = [B_2]_{ОК} + 2_0 = 1{:}0010+1 = 1{:}0011.$

Основні властивості додаткового коду:

– додавання додаткових кодів додатного числа з його від'ємним значенням дає так звану машинну одиницю додаткового коду:

Модк = МОок + 2_0 = 10: 00…00,

тобто число 10 (два) в знакових розрядах числа;

– додатковий код отримав свою назву тому, що зображення від'ємних чисел є доповненням прямого коду чисел до машинної одиниці МОдк.

Модифіковані обернені та *додаткові коди* двійкових чисел відрізняють відповідно від обернених і додаткових кодів подвоєнням значень знакових розрядів. Знак "+" у цих кодах кодують двома нульовими знаковими розрядами, а " -" – двома одиничними розрядами.

Приклад 1.7

$A_{10} = 9$ $A_2 = +1001$ $[A_2]_П = [A_2]_{ОК} = [A_2]_{ДК} = 0{:}1001$

$[A_2]_{МОК} = [A_2]_{МДК} = 00{:}1001;$

$B_{10} = -9$ $B_2 = -1001$ $[B_2]_{ОК} = 1{:}0110$ $[B_2]_{ДК} = 1{:}0111$

$[B_2]_{МОК} = 11{:}0110$ $[B_2]_{МДК} = 11{:}0111.$

Метою введення модифікованих кодів є фіксація та виявлення випадків отримання неправильного результату, коли значення результату перевищує максимально можливий результат у відведеній розрядній сітці комп'ютера.

Структурна схема ЕОМ описує програмний принцип управління, що відповідає послідовному характеру перетворень, що виконується за певним алгоритмом (програмою). Узагальнена структурна схема ЕОМ перших поколінь представлена на рисунку 1.2.

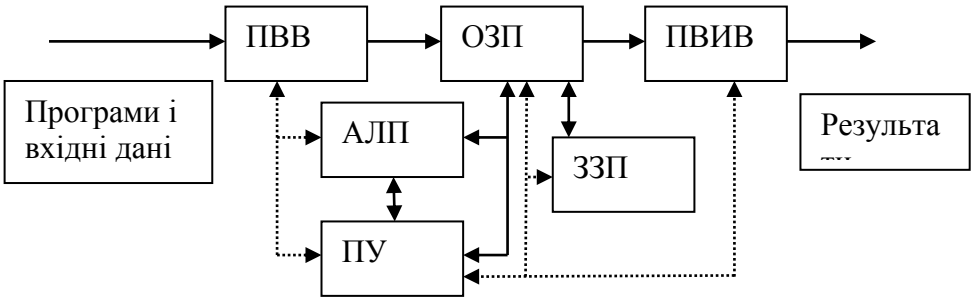

Рисунок 1.2. Структурна схема ЕОМ першого і другого поколінь

Введена інформація, за допомогою пристрої введення інформації (ПВВ), повністю або частково спочатку запам'ятовується в оперативному запам'ятовуючому пристрої (ОЗП), а потім переноситься у зовнішній запам'ятовуючий пристрій (ЗЗП), призначений для тривалого зберігання інформації, де перетворюється в спеціальний програмний об'єкт – файл. Згідно стандарту (ISO 2382/1-84) *файл* – ідентифікована сукупність примірників повністю описаного в конкретній програмі типу даних, що знаходяться поза програмою у зовнішній пам'яті і доступних програмі за допомогою спеціальних операцій.

При використанні файлу в обчислювальному процесі його вміст переноситься в ОЗП. Потім програмна інформація команда за командою зчитується в пристрій управління (ПУ).

Пристрій управління призначається для автоматичного виконання програм шляхом примусової координації всіх інших пристроїв ЕОМ. Ланцюги сигналів управління показані на рис. 2 штриховими лініями. Викликані з ОЗП команди дешифруються пристроєм управління, тобто визначають код операції, яку необхідно виконати наступної, і адреси операндів, які беруть участь у цій операції.

Залежно від кількості використаних в команді операндів розрізняються *одно-, двох-, трехадресні і безадресні команди*. У *одноадресних* командах вказується, де знаходиться один з двох операндів, що обробляються. Другий операнд повинен бути поміщений заздалегідь у арифметичний пристрій (для цього в систему команд вводяться спеціальні команди пересилання даних між пристроями).

Двохадресні команди містять вказівки щодо двох операндів, які розміщуються в пам'яті (або в регістрах і пам'яті). Після виконання команди в одну з цих адрес засилається результат, а той операнд, що знаходився там втрачається.

У *трьохадресних* командах зазвичай дві адреси вказують, де знаходяться вхідні операнди, а третя - куди необхідно помістити результат.

У *безадресних* командах зазвичай обробляється один операнд, який до і після операції перебуває в одному з регістрів арифметико-логічного пристрою (АЛП). Крім того, безадресні команди використовуються для виконання службових операцій (очистити екран, блокування клавіатуру, розблокування і ін.).

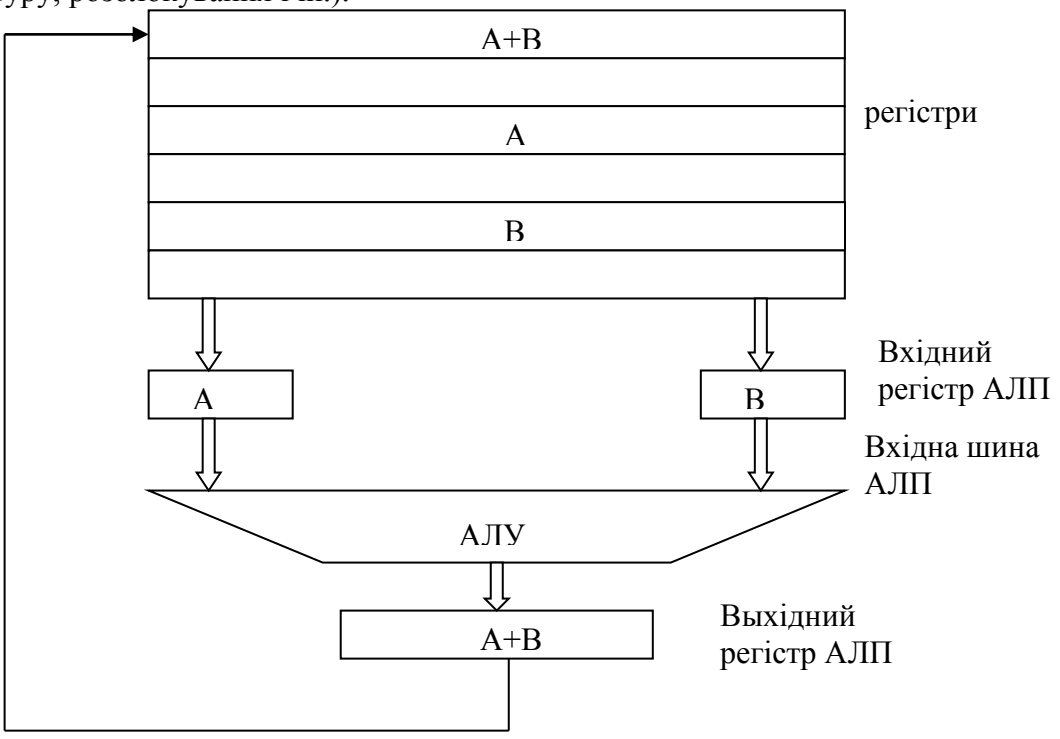

Рисунок 1.3. Схема тракту даних

Всі команди програми виконуються послідовно, команда за командою, в тому порядку, як вони записані в пам’'яті ЕОМ (природний порядок проходження команд). Цей порядок характерний для лінійних програм, тобто програм, що не містять розгалужень. Для організації розгалужень використовуються команди, які порушують природний порядок проходження команд. Пристрій управління використовується для зміни порядку виконання команд програми.

АЛП виконує арифметичні і логічні операції над даними. Основною частиною АЛП є операційний автомат, до складу якого входять суматори, лічильники, регістри, логічні перетворювачі та ін. Він кожен раз переналаштовується на виконання чергової операції.

Результати виконання окремих операцій зберігаються для подальшого використання на одному з регістрів АЛП або записуються в пам'ять. Результати, отримані після виконання всієї програми обчислень, передаються на пристрої виведення (ПВИВ) інформації. Як ПВИВ можуть використовуватися екран дисплея, принтер, плотер і ін. Схема роботи тракту даних представлена на рисунку 1.3.

Регістри разом з АЛП формують тракт даних, по якому надходять дані. Вибирається один або 2-а регістра, АЛП здійснює над ними будь-яку операцію, а результат поміщає в один з цих регістрів.

Робота тракту даних контролюється програмою яка називається мікропрограмою або апаратними засобами.

В ЕОМ третього покоління відбулося ускладнення структури за рахунок розділення процесів введення-виведення інформації та її обробки (рис. 1.4).

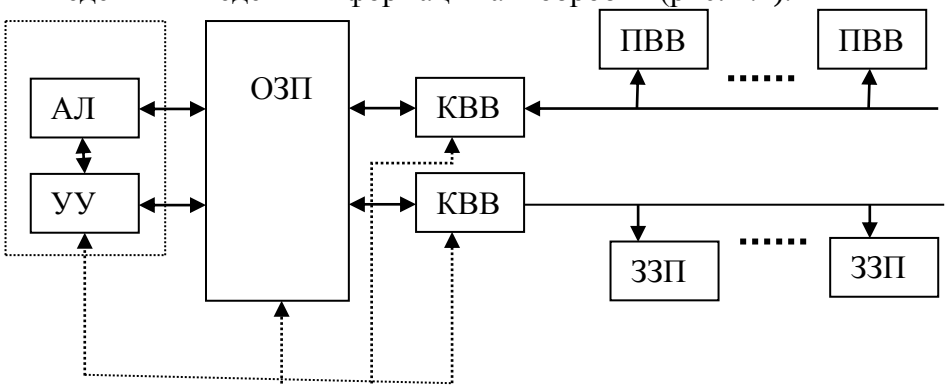

Рисунок 1.4. Структурна схема ЕОМ третього покоління

Сильно пов’язані пристрої АЛП і УУ отримали назву процесор, тобто пристрій, призначений для обробки даних. У схемі ЕОМ з’явилися також додаткові пристрої, які мали назви: процесори введення-виведення, пристрої керування обміном інформацією, канали введення-виведення (КВВ).

Серед каналів вводу-виводу виділяються мультиплексні канали, здатні обслуговувати велику кількість повільно працюючих пристроїв введення-виведення (ПВВ), і селекторні канали, які обслуговували в багатоканальних режимах швидкісні зовнішні запам'ятовуючі пристрої (ЗЗП).

Тут намітилася тенденція до децентралізації управління і паралельної роботи окремих пристроїв, що дозволило різко підвищити швидкодію ЕОМ в цілому.

У персональних ЕОМ, що відносяться до ЕОМ четвертого покоління, відбулася подальше зміна структури (рис. 1.5). Вони успадкували її від міні-ЕОМ.

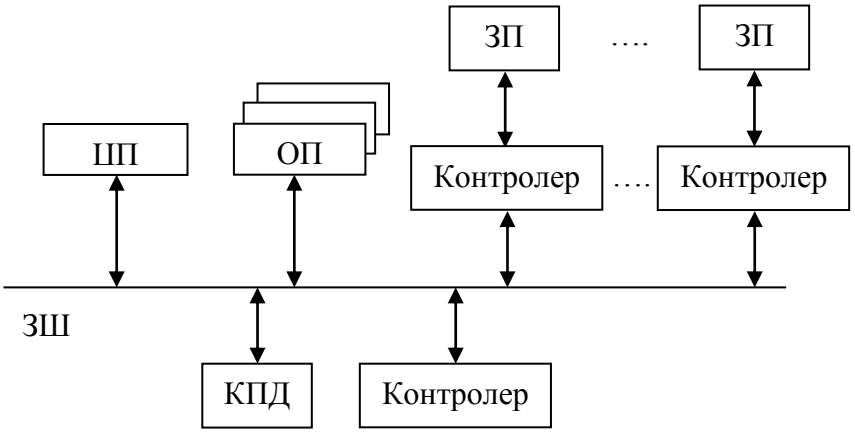

Рисунок 1.5. Структурна схема ПЕОМ

З’єднання всіх пристроїв в єдину машину забезпечується за допомогою загальної шини (ЗШ), що представляє собою лінії передачі даних, адрес, сигналів керування та живлення. Єдина система апаратурних з’єднань значно спростила структуру, зробивши її ще більш децентралізованою. Всі передачі даних по шині здійснюються під керуванням сервісних програм.

Ядро ПЕОМ утворюють центральний процесор (ЦП) і основна пам’ять (ОП). Підключення всіх зовнішніх пристроїв (ЗП), дисплея, клавіатури, дисководів та інших забезпечується через відповідні адаптери – пристрої, які узгоджують швидкості роботи сполучених пристроїв або контролери – спеціальні пристрої управління периферійної апаратурою. Контролери в ПЕОМ грають роль каналів вводу-виводу. В якості особливих пристроїв слід виділити таймер – пристрій вимірювання часу і контролер прямого доступу до пам'яті (КПД) – пристрій, що забезпечує доступ до ОП, минаючи процесор.

У сучасних ЕОМ принципи децентралізації і паралельної роботи поширені як на периферійні пристрої, так і на процесори. Це характерно для обчислювальних систем, що містять кілька обчислювачів (процесорів), які працюють узгоджено і паралельно.

Децентралізація побудови та управління зробила необхідним використовувати модульність побудови і ієрархію управління, які є загальним стандартом структур сучасних ЕОМ.

Модульність побудови передбачає виділення в структурі ЕОМ досить автономних, функціонально і конструктивно закінчених пристроїв (процесор, модулі пам’яті, зовнішні пристрої).

Модульна конструкція ЕОМ робить її відкритою системою, здатною до адаптації і вдосконалення. До ЕОМ можна підключати додаткові пристрої, покращуючи її технічні та економічні показники. З’являється можливість збільшення обчислювальної потужності, поліпшення структури шляхом заміни окремих пристроїв на більш досконалі, зміни та управління конфігурацією системи, пристосування її до конкретних умов застосування відповідно до вимог користувачів. Такий принцип організації називається принципом відкритої архітектури.

Модульність вимагає стандартизації та уніфікації обладнання, номенклатури технічних та програмних засобів, засобів сполучення – інтерфейсів, конструктивних рішень, уніфікації типових елементів заміни, елементної бази та нормативно-технічної документації. Все це сприяє поліпшенню технічних і експлуатаційних характеристик ЕОМ, зростанню технологічності їх виробництва.

Ієрархічна організація структури ЕОМ базується на централізованому управлінні пристрою керування центрального процесора. Підключені до центрального процесора модулі (контролери) можуть, в свою чергу, використовувати спеціальні шини або магістралі для обміну керуючими сигналами, адресами і даними. Ініціалізація роботи модулів забезпечується за командами центральних пристроїв, після чого вони продовжують роботу за власними програмами управління. Результати виконання необхідних операцій транслюються «вгору по ієрархії» для правильної координації всіх робіт.

Ієрархічний принцип побудови і управління характерний не тільки для структури ЕОМ в цілому, але і для окремих її підсистем. Наприклад, за цим же принципом будується система пам’яті ЕОМ.

Так, з точки зору користувача бажано мати оперативну пам’ять великої інформаційної ємності і високої швидкодії. Однак однорівнева побудова пам’яті не дозволяє одночасно задовольняти цим двом суперечливим вимогам. Тому пам’ять сучасних ЕОМ будується за багаторівневим, пірамідальним принципом.

На більш низьких рівнях ієрархії знаходяться зовнішні пристрої. Організація завчасного обміну інформаційними потоками між ЗП різних рівнів при децентралізованому управлінні дозволяє розглядати ієрархію пам’яті як єдину абстрактну (віртуальну) пам’ять. Злагоджена робота всіх рівнів забезпечується під керуванням програм операційної системи.

Децентралізація управління і модульність структури ЕОМ дозволила перейти до більш складних багатопрограмних (мультипрограмних) режимів. При цьому в ЕОМ одночасно може оброблятися кілька програм користувачів.

ЕОМ, що мають один процесор, багатопрограмна обробка відбувається за рахунок переривань. Вона передбачає паралельну роботу окремих пристроїв, задіяних в виконанні різних завдань користувачів. Наприклад, комп'ютер може виконувати роздруківку документів і приймати повідомлення, що надходять по каналах зв'язку, а процесор при цьому може проводити обробку даних, паралельно користувач може слухати музику і т.п.

ЕОМ або обчислювальні системи, що мають кілька процесорів багатопрограмна робота може бути більш глибокою. Автоматичне управління обчисленнями передбачає ускладнення структури за рахунок включення до її складу систем і блоків, які розділяють різні обчислювальні процеси один від одного, що виключає можливість виникнення взаємних перешкод і помилок (системи переривань і пріоритетів, захисту пам'яті). Самостійного значення в обчисленнях вони не мають, але є необхідним елементом структури для забезпечення цих обчислень.

Всі наведені структури не виходять за межі класичної структури фон Неймана. Їх об'єднує такі традиційні ознаки:

– ядро ЕОМ утворює процесор – єдиний обчислювач в структурі, доповнений каналами обміну інформацією і пам'яттю;

– лінійна організація комірок усіх видів пам'яті фіксованого розміру;

– однорівнева адресація комірок пам'яті, стирає відмінності між усіма типами даних;

– внутрішня машинна мова низького рівня, команди якої містять елементарні операції перетворення простих операндів;

– послідовне централізоване управління обчисленнями;

– досить примітивні можливості пристроїв введення-виведення. Незважаючи на всі досягнуті успіхи, класична структура ЕОМ не забезпечує можливостей подальшого збільшення продуктивності. Намітилася криза, обумовлена рядом істотних недоліків: погано розвинені засоби обробки нечислових даних (структури, символи, графічні образи, звук, дуже великі масиви даних і ін.);

– невідповідність машинних операцій операторам мов високого рівня;

– примітивна організація пам'яті ЕОМ;

– низька ефективність ЕОМ при вирішенні завдань, що допускають паралельну обробку і т.п.

Всі ці недоліки призводять до надмірного ускладнення комплексу програмних засобів.

Усі сучасні ЕОМ мають розвинену систему команд. Виконання будь-якої операції ґрунтується на використанні простих мікроопераций типу додавання та зсув. Це дозволяє мати єдиний арифметико-логічний пристрій для виконання будь-яких операцій, пов'язаних з обробкою інформації. Правила додавання двійкових цифр однойменних розрядів з урахуванням можливих перенесень з попереднього розряду подані у табл. 1.1.

Таблиця 1.1 – Правила додавання двійкових цифр

Значення двійкових чисел A і B			Розряд суми Si	Перенесення в наступний розряд Pi
a_i	b_i	P_{i-1}		
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Дії з прямими кодами двійкових чисел під час виконання операцій пов'язані з необхідністю обліку значень знакових розрядів:

– слід окремо обробляти значущі розряди чисел і розряди знака;

– значення розряду знака впливає на алгоритм виконання операції (додавання може замінюватися відніманням, і навпаки).

Операція віднімання здійснюється до операції додавання перетворенням чисел на обернений або додатковий код. Операцію складання виконують відповідно до табл. 1.2.

Таблиця 1.2 – Таблиця перетворення кодів для додаванні

Необхідна операція	Необхідне перетворення
A+B	A+B
A–B	A+(–B)
–A+B	(–A)+B
–A–B	(–A)+(–B)

Дужки вказують на заміну операції віднімання операцією додавання із оберненим або додатковим кодом відповідного числа. Виконуючи додавання цифр, необхідно дотримувати таких правил.

1. Доданки повинні мати однакове число розрядів. Для вирівнювання розрядної сітки доданків можна дописувати незначущі нулі зліва до цілої частини числа та незначущі нулі справа до дробової частини числа.

2. Знакові розряди чисел ураховують у додаванні так само, як і значущі.

3. Необхідні перетворення кодів проводять зі зміною знаків чисел. Приписані незначущі нулі змінюють своє значення у разі перетворень за загальним правилом.

4. Для утворення одиниці перенесення зі старшого знакового розряду, у разі використання ОК, цю одиницю додають з молодшим числовим розрядом. З використанням ДК одиниця перенесення втрачається. Знак результату формується автоматично, результат зображується в тому коді, у якому подані початкові доданки.

У сучасних ЕОМ числа з плаваючою крапкою зберігаються в пам'яті машини, маючи мантису і порядок (характеристику) в прямому коді та нормалізованому вигляді. Усі арифметичні дії над цими числами виконуються так само, як і в тому випадку, коли вони подані в напівлогарифмічному вигляді (мантиса і десятковий порядок) у десятковій системі числення. Порядки і мантиси обробляють відокремлено.

Операцію додавання (віднімання) проводять у такій послідовності.

1. Порівнюють порядки (характеристики) початкових чисел їх відніманням $p = p1\text{-}p2$. Для виконання цієї операції визначається, чи однаковий порядок мають початкові доданки.

2. Якщо різниця порядків дорівнює нулю, то це вказує на те, що однойменні розряди мантис мають однаковий двійковий порядок. В іншому випадку необхідно проводити вирівнювання порядків.

3. Для вирівнювання порядків число з меншим порядком зрушується вправо на різницю порядків *Ар.* Молодші розряди, що виштовхуються при цьому втрачаються.

4. Після вирівнювання порядків мантиси чисел можна додавати (віднімати) залежно від необхідної операції. Операцію віднімання замінюють операцією додавання відповідно до даних табл. 2.3. Дії над доданками проводять в ОК або ДК за загальними правилами.

5. Порядок результату беруть таким, що дорівнює більшому порядку.

6. Якщо мантиса результату не нормалізована, то здійснюють нормалізацію і корекцію значень порядку.

Приклад 1.8 Додати два числа $A_{10} = 7$ $B_{10} = 16$

$A_2 = +11 = +0111$;

$B_2 = +1000 = +10000$.

Спочатку числа мають різну розрядність, необхідно провести вирівнювання розрядної сітки:

$[A_2]_П = [A_2]_{ЗК} = [A_2]_{ДК} = 0{:}\,00111;$

$[B_2]_П = [B_2]_{ЗК} = [B_2]_{ДК} = 0{:}\,10000.$

Додавання в оберненому або додатковому коді дає один і той самий результат

$$\begin{array}{r} 0:00111 \\ +\,0:10000 \\ \hline C_2 = 0:10111 \qquad C_{10} = +23 \end{array}$$

Унаслідок додавання цифр відсутні перенесення в знаковий розряд та із знакового розряду, що указує на отримання правильного результату.

Приклад 1.9 З числа $A_{10} = +16$ відняти $B_{10} = 7$ в ЗК та ДК. Відповідно до табл. 1.2 повинна бути реалізована залежність А+(–В), у якій другий член перетвориться з урахуванням знака.

Унаслідок додавання чисел у ЗК та ДК були отримані перенесення в знаковий розряд та зі знакового розряду. У разі ЗК перенесення зі знакового розряду вимагає додаткового збільшення одиниці молодшого розряду. У разі ДК це перенесення ігнорують.

$$[A_2]_П = 0:10000 \quad [A_2]_{ЗК} = 0:10000 \quad [A_2]_{ДК} = 0:10000$$

$$[B_2]_П = 1:111 = 1:00111 \quad [B_2]_{ЗК} = 1:11000 \quad [B_2]_{ДК} = 1:11001$$

Додавання у ЗК

$$\begin{array}{r} [A_2]_{ЗК} = 0:10000 \\ +\,[B_2]_{ЗК} = 1:11000 \\ \hline 0:01000 \\ +\qquad 1 \\ C_2 = 0:01001 \\ C_{10} = +9 \end{array}$$

Додавання у ДК

$$\begin{array}{r} [A_2]_{ДК} = 0:10000 \\ +\,[B_2]_{ЗК} = 1:11001 \\ \hline 0:01001 \\ \\ C_2 = 0:01001 \\ C_{10} = +9 \end{array}$$

Приклад 1.10 Додати два числа $A_{10} = +1{,}375$; $B_{10} = -0{,}625$.

Переводимо число в двійкову форму 1,375 = ціла частина 2^0, дробова частина $2^{-1}+2^{-2}+2^{-3}= 0{,}5+0{,}25+0{,}125= 1{,}011$

$$A_2 = +1{,}011 = 0{:}\,1011*101; \quad B_2 = -0.101 = -0{:}101*100.$$

У нормалізованому вигляді ці числа матимуть вигляд:

Порядок	Мантиса
$[A_2]$П=0:1	↑0:1011
	знак числа
$[B_2]$П=0:0	↓1:101

1. Віднімаємо порядки $Ep = p1 - p2 = 1 - 0 = 1$. У машині ці дії вимагає операції додавання з перетворенням порядку чисел на додатковий код:

$$\begin{array}{r} p1 = 0:1 \qquad [p1]_{ДК} = 0:1 \\ p2 = 0:0 \quad + \quad [p2]_{ДК} = 0:0 \\ \hline \Delta p = 0:1 \end{array}$$

Визначаємо, що $\Delta p \neq 0$.

2. Порядок першого числа більший за порядок другого числа на одиницю. Необхідне вирівняти порядки.

3. Для вирівнювання порядків необхідно друге число зрушити вправо на один розряд.

$[B_2]_{исх} = 0{:}\,0\;1{:}\,101$

після зрушення

$[B_2]_П$ = 0: 11:0101

$[m_B]_{ДК}$ = 1: 1011

4. Додаємо мантиси.

$[m_A]_{ДК}$ = 0: 1011

+ $[m_B]_{ДК}$ = 1: 1011

$[m_C]_{ДК}$ = 0: 0110

Мантиса числа C – додатна.

5. Порядок числа C дорівнює порядку числа з великим порядком, тобто $p = +1$.

$[C_2]_П$ = 0: 1 0: 0110.

Очевидно, що мантиса результату не нормалізована, оскільки старша цифра мантиси дорівнює нулю.

6. Нормалізуємо результат шляхом зрушення мантиси на один розряд вліво і відповідно віднімаємо зі значення порядку одиницю:

$[C_2]_П$ = 0: 0 0: 110, C_{10} = +0,75.

Операція множення (ділення) чисел з плаваючою крапкою також вимагає різних дій над порядками і мантисами. Алгоритми цих операцій виконують у наступній послідовності.

1. Виконуючи множення (ділення), порядки додають (віднімають) так, як це робиться над числами з фіксованою крапкою.
2. Виконуючи множення (ділення) мантиси перемножують (ділять).
3. Знаки добутку (частки) формують додаванням знакових розрядів співмножників (діленого та дільника). Можливі перенесення зі знакового розряду ігноруються.

Виконати операції множення та ділення чисел з фіксованою крапкою, а також операцію додавання з плаваючою крапкою.

Приклад 1.11 Помножити два числа A_{10} = 7 B_{10} = 5.

Перемножимо ці числа, представлені прямими двійковими кодами.

$A_2 = 111$ $B_2 = 101$

```
      111
   x  101
---------
      111
     000
    111
---------
   100011
```

Очевидно, що добуток отримають додаванням часткових добутків, що є розрядами множника, зрушеними вліво відповідно до позицій розрядів множника. Частки добутку, отримані множенням на нуль, іґноруються.

Приклад 1.12 Розділити два числа A_{10} = 45 B_{10} = 5

$[A_2]_П$ = 101101 $[B_2]_П$ = 101

```
Ділене          Дільник
 101101    |  101
- 101       1001  – частка
   0101
  - 101
      0
```

$[C_2]_П$ = 1001 C_{10} = 9

Ділення виконують так само, як це робиться зазвичай у десятковій системі. Спочатку перевіряють, чи можна відняти значення дільника зі старших розрядів діленого. Якщо можливо, то в розряді частки записують одиницю і визначають різницю частки. Інакше в

частку записують нуль і розряди дільника зсуву вправо на один розряд відносно розрядів діленого. До отриманої попередньої різниці зносять наступну цифру діленого, і процес повторюється, поки не буде отримана необхідна точність.

ЗАВДАННЯ

1. Перевести числа 128; 113; 29; 1121 у двійкову, вісімкову і шістнадцяткову системи числення.
2. Записати прямий, обернений і додатковий код для чисел 54;–123; 16; –9; 8; –6.
3. Виконати операції додавання у двійкових кодах для чисел +110101 та +101; 100011 та –111; –111111 та –1011.
4. Перемножити двійкові числа і перевірити результат: 1100 і 1011; 101101 і 10; 1001 і 101; 1000111 і 11; 111 і 1000.
5. Розділити двійкові числа і перевірити результат: 1100 і 10; 11110 і 110; 1001000 і 1000; 1100100 і 11001.
6. Перевести числа у двійкову систему числення і додати їх: +0,256 і –0,125; +1,25 і +1,2; –4,11 і –0,025.

КОНТРОЛЬНІ ПИТАННЯ

1. Що таке операнди?
2. Для чого произначені адреси?
3. З яких полів складається кожна команда?
4. Які бувають команди за адресацію?
5. Перерахуйте основні властивості оберненого коду.
6. Перерахуйте основні властивості додаткового коду.
7. Як виконують операцію зсуву під час додавання чисел у двійкових кодах?
8. Як формується знак частки під час множення та ділення двійкових чисел?
9. Який алгоритм множення двійкових чисел з фіксованою крапкою?
10. Чим відрізняється алгоритм множення двійкових чисел з фіксованою крапкою від алгоритму ділення?
11. Яка послідовність виконання додавання і віднімання двійкових чисел з плаваючою крапкою?

РОЗДІЛ 2. ОРГАНІЗАЦІЯ КОМП'ЮТЕРНИХ СИСТЕМ, ЯК КОМПЛЕКСУ ТЕХНІЧНИХ ЗАСОБІВ

2.1. Принципи побудови комп'ютерів.

1. Всі команди безпосередньо виконуються апаратним забезпеченням. Вони не інтерпретуються мікрокомандами. Усунення рівня інтерпретації забезпечує високу швидкість виконання більшості команд.
2. Комп'ютер виконує велику кількість команд завдяки зверненню до великої кількості команд в секунду. Технологія розпаралелювання.
3. Команди легко декодуються. Декодування команд здійснюється для того, щоб визначити, які ресурси їм необхідні і які дії потрібно виконувати.
4. До пам'яті звертаються тільки команди завантаження і збереження. Спосіб розбиття операцій й операндів, які для більшості команд беруться і повертаються в регістри. Операція

переміщення операндів з пам'яті в регістри може здійснюватися в різних командах. Роботу цих команд можуть виконувати інші команди (завантаження і збереження.)

5. Задіяна велика кількість регістрів, якщо слово викликано з пам'яті, воно може знаходитися в регістрі стільки, скільки потрібно.

RISC – Reduced Instruction Set Computer – комп'ютер зі скороченим набором команд (близько 50)

CISC – Complex Instruction Set Computer – комп'ютер з повним набором команд (200-300).

На сьогоднішній день головна відмінність RISC і CISC технологій складається тільки з кількості команд, але назва зберігається і досі. Технологія RISC заснована на тому, що включає невелику кількість простих команд, кожна з яких виконується за один цикл тракту даних, за рахунок цього команда виконується в 10 разів швидше, ніж команда CISC, що складається з безлічі команд.

Зараз використовують комбіновану технологію, процесори містять ядро RISC, яке виконує найпростіші (найпоширеніші) команди, а за звичайною технологією CISC інтерпретуються більш складні команди.

Паралелізм – можливість виконувати дві або більше операцій одночасно. Існують дві основні форми паралелізму – на рівні команд і на рівні процесора.

Конвеєр було знайдено в 1959 році, це дозволило вирішити головну перешкоду високої швидкості – виклику команд з пам'яті. Ідея конвеєра полягала в наступному, команда поділялась на кілька етапів, кожен виконувався певною частиною апаратного забезпечення, всі частини могли працювати паралельно (рис. 2.1). Етапи роботи називалися стадіями

Рисунок 2.1. Конвеєр з 5-ти стадій

Вибірка команди відбувається з пам'яті, потім команда поміщається в буфер. Під час декодування команди визначається її тип і тип операндів. Вибірка операндів визначає місцезнаходження операндів і виклик їх з реєстру або з пам'яті. Виконння команди відбувається шляхом проведення операндів через тракт даних. В блоці повернення результат записується в потрібний регістр.

Конвеєр дозволяє знайти компроміс між часом очікування (час виконання однією команди) і пропускною спроможністю. Якщо час циклу становить T нс, а конвеєр містить n стадій, то час очікування складе nT нс, а пропускна здатність - $1000/T$ млн команд в секунду.

Всі команди, що виконує процес поділяються на два типи: регістр-пам'ять, регистр-регистр.

Процесор виконує кожну команду за кілька кроків:

1. Викликає наступну команду з пам'яті і переносить її в регістр команд;
2. Змінює положення лічильника команд, який тепер повинен вказувати на наступну команду;
3. Визначає тип викликаної команди;
4. Якщо команда використовує слово пам'яті визначає, де знаходиться слово;
5. Переносить слово, якщо це необхідно, в регістр центрального процесора;
6. Виконує команду;
7. Переходить до кроку 1.

Така послідовність кроків (вибірка – декодування – виконання) є основною роботою процесора.

Використання двох конвеєрів. Кожен конвеєр містить АЛП для паралельних операцій. Ця технологія була реалізова в Pentium. Особливі компілятори об'єднували сумісні команди в пари, за рахунок чого швидкість обробки цілих чисел збільшилася у два рази.

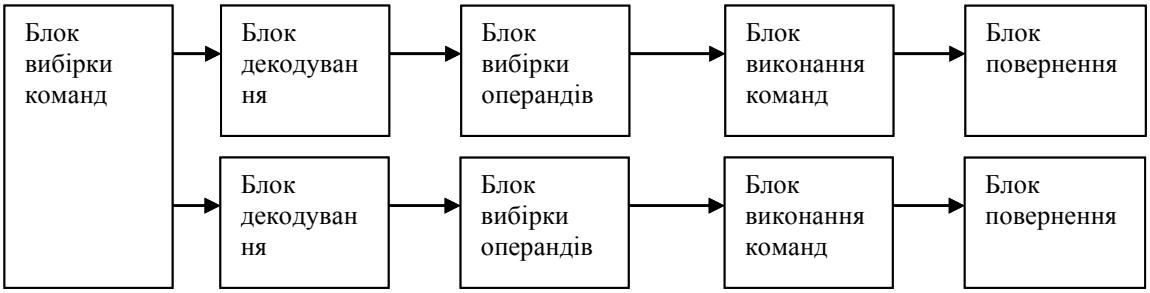

Рисунок 2.2. Два конвеєра

В Pentium II реалізована суперскалярна архітектура (1987 рік), в якій використовується один конвеєр з великою кількістю функціональних блоків.

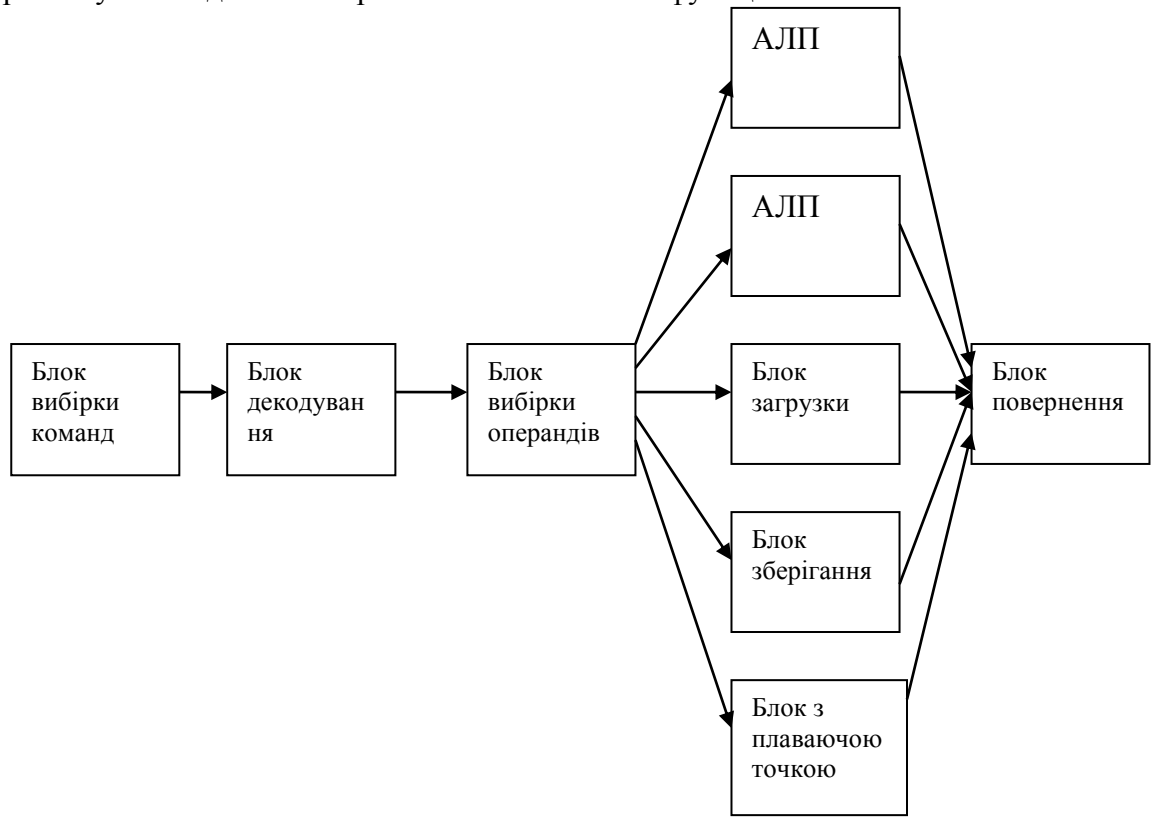

Рисунок 2.3. Суперскалярна архітектура

Багатоядерний процесор складається з двох і більше обчислювальних ядер на одному кристалі. Він має один корпус і встановлюється в один роз'єм на системній платі комп'ютера при цьому операційна система сприймає кожне його обчислювальне ядро як окремий процесор з повним набором обчислювальних ресурсів. Це забезпечує паралелізм на рівні процесів і нитей, що збільшує продуктивність багатозадачної роботи на оптимізованому під таку роботу програмному забезпеченні.

2.2. Паралелізм на рівні процесорів

Багатопроцесорні системи є кількох типів в залежності від організації ядра комп'ютера, наприклад:

– Масивно-паралельний процесор складається з великого числа подібних процесорів, які виконують одну й ту ж послідовність команд стосовно до різних наборів даних. Решітка

8x8 елементів процесор/пам’ять з одним блоком контролю для кожного сектора, який розсилав команди і вони виконувалися усіма процесорами одночасно.

– Векторний процесор має один блок додавання, який побудований за конвеєрної структурою, містить векторний регістр, що складається з набору стандартних регістрів і в якому виконуються всі операції додавання.

– Мультіпроцесорна система побудована з декількох паралельних процесорів, які поділяють одну загальну пам’ять.

Багатопроцесорні системи це суто професійне, вузькоспеціалізоване рішення, що застосовується для створення серверів, потужних графічних станцій і кластерів в суперкомп’ютерах.

2.3. Побудова процесора.

Існує обов’язковий (стандартний) мінімальний набір функціональних засобів для кожного типу центрального процесора. Він включає в себе: регістри загального призначення, засоби виконання стандартного набору операцій і засоби управління обчислювальним процесом. Реалізація окремого ЦП може відрізнятися складом засобів, способом їх реалізації, технічними параметрами.

Умовно мікропроцесор можна розділити на дві частини: виконавчий блок (Execution Unit - EU) і пристрій сполучення з системною магістраллю (Bus Interface Unit - BIU).

У виконавчому блоці знаходяться: арифметичний блок і регістри загального призначення (РЗП). Арифметичний блок включає арифметико-логічний пристрій, допоміжні регістри для зберігання операндів і регістр прапорів.

Всі регістри виконавчого блоку МП (AX, BX, CX, DX, SP, BP, SI, DI), що мають довжину машинного слова й поділяється на дві групи. Першу групу складають регістри загального призначення: AX, BX, CX і DX. Другу групу складають адресні регістри SP, BP, SI і DI (в старших моделях кількість адресних регістрів збільшено).

Пристрій сполучення з системною магістраллю містить керуючі регістри, конвеєр команд, АЛП команд, пристрій управління виконавчим блоком МП і інтерфейс пам'яті (що з'єднує внутрішню магістраль МП з системною магістраллю ПЕОМ).

Керуючі регістри BIU: CS (покажчик командного сегмента), DS покажчик сегмента даних), SS (покажчик сегмента стека), ES (покажчик додаткового сегмента) і ін. Служать для визначення фізичних адрес ОП - операндів і команд. Регістр IP (Instruction Pointer) – лічильник команд є покажчиком адреси команди, яка буде вибиратися з конвеєру команд в якості чергової команди. Конвеєр команд МП зберігає кілька команд, що дозволяє при виконанні лінійних програм поєднати підготовку чергової команди з виконанням поточної.

До керуючих регістрів МП відноситься і регістр прапорів, кожен розряд якого має строго визначене призначення. Зазвичай розряди регістра прапорів встановлюються апаратно при виконанні чергової операції в залежності від результату, який одержано в АЛП. При цьому фіксуються такі властивості одержуваного результату: нульовий результат, від’ємне число, переповнення розрядної сітки АЛП і т.д. Але деякі розряди регістра прапорів можуть встановлюватися за спеціальними командами. Деякі розряди мають чисто службове призначення (зберігають розряд, який «випав з АЛП» під час зсуву або є зарезервованими (тобто не використовуються).

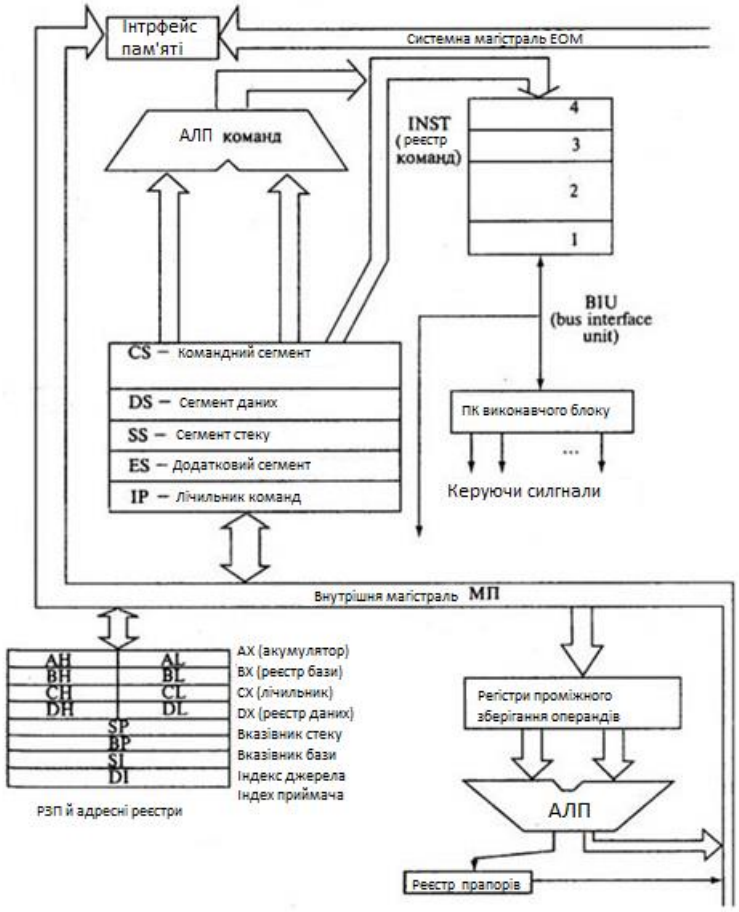

Рисунок 2.4. Структурна схема процесора.

Всі прапори молодшого байта регістра встановлюються арифметичними або логічними операціями МП. Всі прапори старших байтів, за винятком прапора переповнення, встановлюються програмним шляхом, для цього в МП є команди установки прапорів (STC, STD, STI), скидання (CLC CLD, CLI), інвертування (CMC).

2.4. Взаємодія елементів при роботі мікропроцесора

Роботою МП управляє програма, записана в ОП ЕОМ. Адреса чергової команди зберігається в лічильнику команд IP (Instruction Pointer) і в одному з сегментних регістрів, найчастіше в CS. У реальному режимі вся ОП ділиться на сегменти (довжина сегмента – 64 Кбайта). Адреса ОП поділяється на дві частини: номер сегмента в ОП (база сегмента) і номер комірки всередині даного сегмента (зміщення відносно початку сегмента). Базова адреса сегмента утворюється додаванням до номера сегмента справа чотирьох нулів. Оскільки останні чотири розряди абсолютної (фізичної) адреси сегмента завжди нульові, сегмент може починатися не з будь-якої комірки ОП, а тільки з "параграфа" - початку блоку ОП. У структурі мікропроцесора є кілька регістрів сегментів, наприклад в комп’ютерах молодших поколінь їх чотири:

- CS – програмний сегмент;
- DS – сегмент даних (інформаційний сегмент);
- SS – сегмент стеку;
- ES – розширений сегмент (додатковий сегмент даних).

Фізична адреса чергової команди через внутрішню магістраль МП і інтерфейс пам’яті надходить до шини адреси системної магістралі. Одночасно з пристрою керування (ПК) виконавчого блоку на шину керування видається команда (керуючий сигнал) в ОП, відповідно до якої треба вибрати число, що знаходиться за адресою, вказаною в системній

магістралі. Обране число, яке є черговою командою, надходить з ОП через шину даних системної магістралі, інтерфейс пам’яті, внутрішню магістраль МП на регістр команд (INST).

З команди до реєстру команд надходить код операції з пристрою керування виконавчого блоку для вироблення керуючих сигналів, які налаштовують мікропроцесор на виконання необхідної операції.

Залежно від режиму адресації, який використовується в команді, організується вибірка необхідних вихідних даних.

2.5. Склад та принцип дії основної пам’яті

Комплекс технічних засобів, що реалізують функцію пам’яті, називається запам’ятовуючими пристроями (ЗП). ЗП необхідні для розміщення в них команд і даних. Вони забезпечують центральному процесору доступ до програм та інформації.

Пристрої, що запам’ятовують діляться на:

- основну пам’ять,
- надоперативну пам’ять (НОЗП)
- зовнішні запам’ятовуючи пристрої.

Основна пам’ять включає в себе два типи пристроїв: оперативний запам’ятовуючий пристрій (ОЗП або RAM - Random Access Memory) і постійний запам’ятовуючий пристрій (ПЗП або ROM - Read Only Memory).

ОЗП призначений для зберігання змінної інформації. Він допускає зміну свого вмісту в ході виконання процесором обчислювальних операцій з даними і може працювати в режимах запису, читання, зберігання.

ПЗП містить інформацію, яка не повинна змінюватися в ході виконання процесором обчислювальних операцій, наприклад стандартні програми і константи. Ця інформація заноситься в ПЗП перед установленням мікросхеми в ЕОМ. Основними операціями, які може виконувати ПЗП, є читання і зберігання.

Функціональні можливості ОЗП ширше, ніж ПЗП. Але ПЗП зберігає інформацію при відключенні живлення (тобто є енергонезалежною пам’яттю) і може мати більш високу швидкодію, так як обмеженість функціональних можливостей ПЗП і його спеціалізація на читанні та зберіганні дозволяють скоротити час виконання реалізованих їм операцій зчитування.

У сучасних ЕОМ мікросхеми пам’яті виготовляють з кремнію по напівпровідникової технології з високим ступенем інтеграції елементів на кристалі (мікросхеми пам’яті відносяться до так званих «регулярних» схем, що дозволяє зробити установку елементів пам'яті в кристалі (чіпі) настільки щільною, що розміри елементів пам’яті стають порівнянними з розмірами окремих атомів).

Основною складовою частиною мікросхеми є масив елементів пам’яті (ЕП), об’єднаних в матрицю накопичувача. Кожен елемент пам’яті може зберігати 1 біт інформації і має свою адресу. ЗП, що дозволяють звертатися за адресою до будь-якого ЕП в довільному порядку, називаються запам’ятовуючими пристроями з довільним доступом.

При матричної організації пам’яті реалізується координатний принцип адресації ЕП, в зв’язку з чим адреса ділиться на дві частини (дві координати) – X і Y. На перетині цих координат знаходиться елемент пам’яті, інформація якого повинна бути прочитана або змінена.

ОЗП пов’язаний з іншим мікропроцесорним комплектом ЕОМ через системну магістраль (СМ), яка складається з шини управління, шини даних і шини адрес. По шині управління передається сигнал, що визначає, яку операцію необхідно виконати. По шині даних передається інформація, що записується в пам’ять або зчитується з неї. По шині адреси передається адреси елементів пам’яті, які беруть участь в обміні (оскільки дані передаються машинними словами, а один ЕП може сприйняти тільки один біт інформації,

блок елементів пам’яті складається з *n* матриць ЕП, де *n* – кількість розрядів в машинному слові). Максимальна ємність пам’яті визначається кількістю ліній в шині адреси системної магістралі: якщо кількість ліній позначити *m*, то ємність пам’яті (тобто кількість елементів пам’яті, що мають унікальні адреси) визначається як 2*m*.

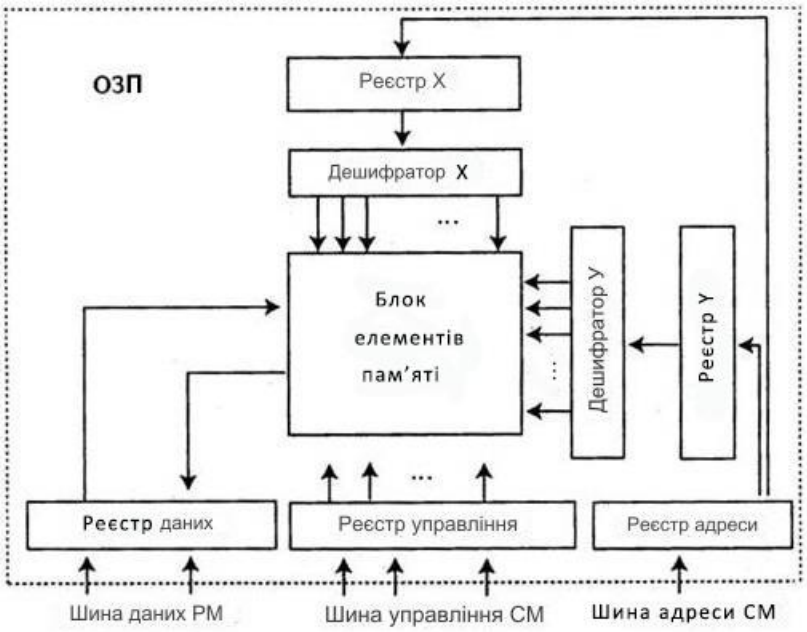

Рисунок 2.5. Структурна схема ОЗУ.

Мікросхеми пам’яті можуть будуватися на статичних (SRAM) і динамічних (DRAM) ЕП. У якості статичного ЕП найчастіше виступає статичний тригер. У якості динамічного ЕП може використовуватися електричний конденсатор сформований всередині кремнієвого кристала.

Статичні ЕП здатні зберігати свій стан (0 або 1) необмежено довго (при увімкненому живленні). Динамічні ЕП з плином часу записану в них інформацію втрачають (наприклад, через саморозряду конденсатора), тому вони потребують періодичного відновлення записаної в них інформації – регенерації.

Мікросхеми елементів пам’яті динамічних ОЗП відрізняються від аналогічних ЕП статичних ОЗП меншим числом компонентів в одному елементі пам’яті, у зв’язку з чим мають менші розміри і можуть бути більш щільно упаковані в кристалі. Однак через необхідність регенерації інформації динамічні ОЗП мають більш складні схеми керування.

Основними характеристиками ОЗП є обсяг і швидкодія.

В сучасних ПЕОМ ОЗП має модульну структуру. Змінні модулі можуть мати різне конструктивне виконання (SIP, ZIP, SIMM, DIMM). Збільшення обсягу ОЗП зазвичай пов'язане з установкою додаткових модулів, які випускаються в 30-контактному (30-pin) і 72-контактному виконаннях на 8, 16, 32 і 64 Мбайт. Час доступу до модулів DRAM становить 60 – 70 нс.

На продуктивність ЕОМ впливає не тільки час доступу, але і такі параметри (пов'язані з ОЗП), як тактова частота і розрядність шини даних системної магістралі. Якщо тактова частота недостатньо висока, то ОЗП простоює в очікуванні звернення. При тактовій частоті, що перевищує можливості ОЗП, в очікуванні буде знаходитися системна магістраль, через яку надійшов запит до ОЗП.

Розрядність шини даних (8, 16, 32 або 64 біта) визначає довжину інформаційної одиниці, якій можна обмінятися з ОЗП за одне звернення.

Інтегральною характеристикою продуктивності ОЗП з урахуванням частоти і розрядності є пропускна здатність, яка вимірюється в Мегабайтах в секунду.

Мікросхеми ПЗП також побудовані за принципом матричної структури накопичувача. Функції елементів пам'яті в них виконують перемички у вигляді провідників, напівпровідникових діодів або транзисторів. У такій матриці наявність перемички може означати «1», а її відсутність – «0». Занесення інформації в мікросхему ПЗП називається її програмуванням, а пристрій, за допомогою якого заноситься інформація – програматором. Програмування ПЗП полягає в видаленні (пропалюванні) перемичок за тими адресами, де повинен зберігатися «0». Зазвичай схеми ПЗП допускають тільки одне програмування, але спеціальні мікросхеми, які перепрограмують допускають багаторазове стирання і занесення нової інформації. Цей вид мікросхем також відноситься до енергозалежних, тобто може тривалий час зберігати інформацію при вимкненому живленні (стирання мікросхеми відбувається або за рахунок подачі спеціальної напруги, що стирає, або за рахунок впливу на кристал ультрафіолетового випромінювання, для цього в корпусі мікросхеми залишається прозоре вікно).

Надоперативні ЗП використовуються для зберігання невеликих обсягів інформації та мають значно менший час (в 2 - 10 разів) зчитування/запису, ніж основна пам'ять. НОЗП зазвичай будуються на реєстрах і реєстрових структурах.

Реєстр являє собою електронний пристрій, здатний зберігати занесене в нього число необмежено довго (при увімкненому живленні). Найбільшого поширення набули реєстри на статичних тригерах.

За призначенням реєстри діляться на реєстри зберігання і реєстри зсуву. Інформація в реєстри може заноситися і зчитуватися або паралельно, одразу всіма розрядами, або послідовно, через один з крайніх розрядів з подальшим зрушенням занесеної інформації.

Зрушення записаної в реєстр інформації може проводитися вправо або вліво. Якщо реєстр допускає зрушення інформації в будь-якому напрямку, він називається *реверсивним*.

Реєстри можуть бути об'єднані в єдину структуру. Можливості такої структури визначаються способом доступу і адресації реєстрів.

Якщо до будь-якого реєстру можна звернутися для запису/читання на його адресу, така реєстрова структура утворює *НОЗП* (*кеш-пам'ять*) *з довільним доступом*.

Безадресні реєстрові структури можуть утворювати два види пристроїв пам'яті: магазинного типу і пам'ять з вибіркою за змістом (асоціативні ЗП).

Пам'ять магазинного типу утворюється з послідовно з'єднаних реєстрів (рис. 2.6). Якщо запис в реєстрову структуру (рис. 2.6. а) проводиться через один реєстр, а зчитування – через інший, то така пам'ять є аналогом лінії затримки і працює за принципом «першим увійшов – першим вийшов» (FIFO – first input, first output).

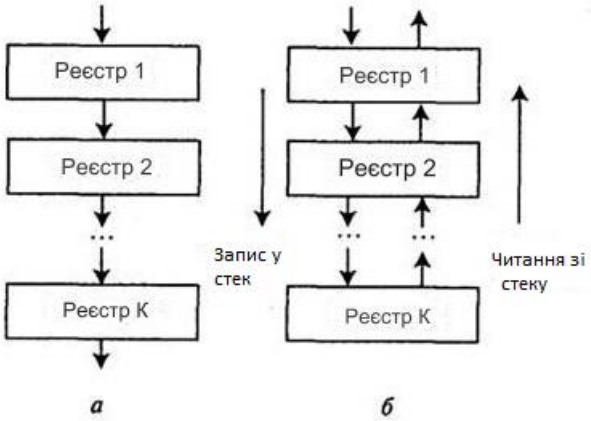

Рисунок 2.6. Реєстрова структура магазинного типу: а - типу FIFO; б - типу FILO

Якщо ж запис і читання здійснюються через один і той же регістр (рис. 2.6. б), такий пристрій називається *стековою пам'яттю*, що працює за принципом «першим увійшов – останнім вийшов» (FILO – first input, last output). При запису числа в стекову пам'ять спочатку вміст стека зміщується в бік останнього, k-го регістра (якщо стек був повністю

заповнений, то число з k-го регістра втрачається), а потім число заноситься в вершину стека – регістр 1. Читання здійснюється теж через вершину стека, після того як число з вершини прочитано, стек зсувається в бік регістра 1.

Стекова пам'ять набула широкого поширення. Для її реалізації в ЕОМ розроблені спеціальні мікросхеми, також робота стекової пам'яті емулюється в ОП ЕОМ за допомогою програм операційної системи, які виділяють частину пам'яті під стек. Спеціальний реєстр мікропроцесора (покажчик стека) постійно зберігає адресу комірки ОП, яка виконує функції вершини стека. Читання числа завжди проводиться з вершини стека, після чого покажчик стека змінюється і вказує на чергову комірку стекової пам'яті (тобто фактично стек залишається нерухомим, а переміщається вершина стека). При запису числа в стек спочатку номер комірки в покажчику стека модифікується так, щоб він вказував на чергову вільну комірку, після чого проводиться запис числа за цією адресою. Така робота покажчика стека дозволяє реалізувати принцип «першим увійшов – останнім вийшов». До стеку може бути завантажений в певній послідовності ряд даних, які згодом зчитуються зі стека вже в зворотному порядку, на цій властивості побудована система арифметичних перетворень інформації, відома під назвою «логіка Лукашевича».

Пам'ять з вибіркою за змістом є безадресної. Звернення до неї здійснюється за спеціальною маскою, яка містить пошуковий образ. Інформація зчитується з пам'яті, якщо частина її відповідає пошуковому образу, зафіксованому в масці. Наприклад, якщо в таку пам'ять записана інформація, що містить дані про місце проживання (включно з містом), і необхідно знайти відомості про жителів певного міста, то назва цього міста поміщається в маску і дається команда читання, яка вибирає з пам'яті всі записи, що стосуються заданого міста.

У мікропроцесорах асоціативні ЗП використовуються в складі кеш-пам'яті для зберігання адресної частини команд і операндів виконуваної програми. При цьому немає необхідності звертатися до ОП, досить помістити в маску потрібну адресу, якщо шукана інформація є в кеш-пам'яті, то вона буде відразу буде знайдена. Звернення до ОП необхідно лише при відсутності необхідної інформації в НОЗП. За рахунок такого використання НОЗП скорочується число звернень до ОП, а це дозволяє економити час, так як звернення до НОЗП вимагає в 2 - 10 разів менше часу, ніж звернення до ОП.

НОЗП (кеш-пам'ять) може бути розміщена в кристалі процесора (так звана «кеш-пам'ять I рівня») або виконана у вигляді окремої мікросхеми (зовнішня кеш-пам'ять або кеш-пам'ять II рівня і т.д.).

2.6. Розміщення інформації в основний пам'яті ЕОМ

Кожен байт, записаний в ОП, має унікальний номер (адресу). При використанні 20-бітної шини адреси абсолютна (фізична) адреса кожного байта є п'ятирозрядне шістнадцяткове число, яке приймає значення від 00000 до FFFFF. У молодших адресах розташовуються блоки операційної системи (вектори переривань, зарезервована область пам'яті BIOS), в цій же частині можуть розміщуватися драйвери пристроїв, додаткові обробники переривань, командний процесор операційної системи. Потім розташовується область пам'яті, відведена користувачеві. Запис в ОП (і читання з неї) може здійснюватися не тільки байтами, але і машинними словами. При цьому машинне слово при розміщенні в пам'яті займає кілька суміжних байтів. Але машинне слово характеризується не всіма адресами зайнятих байтів, а тільки одним – адресою молодшого байта слова.

Рисунок 2.7. Стереотипне уявлення машинного слова

При запису слова молодший байт розміщується за адресою, яка є адресою машинного слова, старший байт машинного слова розміщується в наступному порядку байті ОП, що має номер, збільшений на 1 (тут діє мнемонічне правило «молодший байт – за молодшою адресою»).

При читанні з ОП двох наступних байтів машинного слова їх прийнято розміщувати зліва направо: спочатку перший з прочитаних байтів (з меншою адресою), а потім – наступний. В результаті відбувається «обертання» байтів (рис. 2.7), яке психологічно важко сприймається. Необхідно пам’ятати, що під час запису окремих байтів кожен байт розташовується в ОП за своєю адресою, при читанні ніякого обертання не відбувається. При запису в ОП одиниць інформації, що мають в своєму складі більше одного байта, адресою інформаційної одиниці є адреса самого молодшого байта, запис в ОП ведеться побайтно, починаючи з самого молодшого байта, кожен наступний байт розташовується в комірці, адреса якої на 1 більше попередньої . Іншими словами, запис машинного чи подвійного слова проводиться справа наліво, тоді як при читанні лічені байти зазвичай розташовуються зліва направо – відбувається «обертання» байтів, перестановка їх місцями, що необхідно враховувати при роботі з ОП на фізичному рівні.

Мінімальний розмір пам’яті, що виділяється даним – сторінка, що дорівнює 1000 байтам (4096 байт – 4КБ). Такий метод розподілу пам’яті називається гранулярністю. Це потрібно під час роботи з файлами для того, щоб з цієї пам’яті читати файл.

Можна отримати значення з пам’яті та помістити в регістр. Для прикладу візьмемо таку схему пам’яті, де кожен блок є байтом

зсув	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40	41	42
дані	0D	0A	50	32	44	57	25	7A	5E	72	EF	7D	FF	AD	C7

Значення зсуву позначене тут як байт, але насправді це – 32-розрядне значення. Візьмемо для прикладу 3A, це також 32-розрядне значення: 0000003Ah. Щоб заощадити простір, використовують маленькі зсуви.

Дані на зсуві 3A – 25, 7A, 5E, 72, EF, і т. д. Наприклад, щоб помістити значення зі зсуву 3A, з розміром DWORD (32-біта) з пам'яті зі зсувом 3Ah у регістр EAX. Після цього, EAX міститиме значення 725E7A25h. Це – інверсія того, що знаходиться у пам'яті: 25 7A 5E 72. Значення зберігаються у пам'яті, використовуючи формат little endian. Це означає, що наймолодший байт зберігається у найбільш значущому байті, порядок байтів ззаду наперед.

Наприклад:

- dword (32-біт) шістнадцяткові значення 10203040 зберігаються у пам’яті як: 40, 30, 20, 10
- word (16-біт) шістнадцяткові значення 4050 зберігається у пам’яті як: 50, 40

2.7. Принципи організації підсистеми введення-виведення

Продуктивність і ефективність використання ЕОМ визначаються не тільки можливостями її процесора і характеристиками основної пам’яті, але в дуже великій мірі складом її периферійних пристроїв (ПП), їх технічними даними і способом організації їх спільної роботи з ядром (процесором і основною пам’яттю) ЕОМ.

Зв’язок пристроїв ЕОМ один з одним здійснюється за допомогою інтерфейсів.

Інтерфейс являє собою сукупність ліній і шин, сигналів, електронних схем і алгоритмів (протоколів), призначену для здійснення обміну інформацією між пристроями. Від характеристик інтерфейсів багато в чому залежать продуктивність і надійність ЕОМ.

Основу характеристик інтерфейсів складає *модульність*, яка передбачає що окремі пристрої виконуються у вигляді конструктивно закінчених модулів. При цьому приєднання нового пристрою не повинно викликати в існуючій частині машини ніяких змін, крім зміни кабельних з’єднань і деяких коригувань програм.

Важливим чинником є *уніфіковані формати даних* (які не залежать від типу ПП), ними ПП обмінюються з ядром ЕОМ, в тому числі уніфікований формат повідомлення, яке ПП посилає в ядро про свій стан. Перетворення уніфікованих форматів даних в

індивідуальні, якщо це необхідно, проводиться в самих ПП, точніше, в блоках управління ПП (контролерах, адаптерах).

Ще один важливий чинник це *уніфікований інтерфейс*, тобто уніфікований за складом і призначенням набір ліній і шин, уніфіковані схеми підключення, сигнали і алгоритми (протоколи) управління обміном інформацією між ПП і ядром ЕОМ. Уніфікований (які не залежать від типу ПП) формат і набір команд процесора для операцій введення-виведення. Операція введення-виведення з будь яким ПП представляє для процесора просто операцію передачі даних незалежно від особливостей принципу дії даного ПП, типу його носія і т.п.

Для забезпечення паралельної в часі роботи ПП з виконанням процесором програми, схеми управління введенням-виведенням відокремлюють від процесора і надають їм достатній ступінь автономності. Багато функцій управління операціями введення-виведення, як наприклад управління прямим доступом до пам’яті, є загальними, вони не залежать від типу ПП.

Виконання загальних функцій покладають на загальні для груп ПП уніфіковані пристрої - контролери прямого доступу до пам’яті, процесори (канали) введення-виведення, а специфічних - на спеціалізовані для даного типу ПП електронні блоки управління (адаптери).

У загальному випадку для організації та проведення обміну даними між двома пристроями потрібні такі спеціальні засоби:

- спеціальні керуючі сигнали і їх послідовності;
- пристрої сполучення;
- лінії зв’язку;
- програми, що реалізують обмін.

Класифікація інтерфейсів за призначенням відображає взаємозв’язок з архітектурою реальних засобів обчислювальної техніки. Відповідно до цієї ознаки в ЕОМ і обчислювальних системах можна виділити кілька рівнів сполучення:

- машинні системні інтерфейси;
- інтерфейси периферійних пристроїв або інтерфейси введення-виведення (малі інтерфейси);
- міжмашинні інтерфейси;
- інтерфейси «людина – машина» – для обміну інформацією між людиною і ЕОМ.

Машинні (внутрішньо машинні) системні інтерфейси є базовою часткою архітектури ЕОМ і сукупністю уніфікованої магістралі, електронних схем, які управляють проходженням сигналів по шинах і т.п. Призначені для організації зв’язків між складовими компонентами ЕОМ на рівні обміну інформацією з центральним процесором, ОП і контролерами (адаптерами) ПП.

Є два типи шин – системна і локальна.

Системна шина призначена для забезпечення передачі даних між периферійними пристроями і центральним процесором, а також оперативною пам’яттю.

Локальною шиною, як правило, називається шина, яка безпосередньо підключена до контактів мікропроцесора, тобто шина процесора, яка за допомогою електричних сигналів виходить безпосередньо на контакти мікропроцесора, і призначена для збільшення швидкодії відеоадаптерів і контролерів дискових накопичувачів. Вона зазвичай об’єднує процесор, пам’ять, схеми буферизації для системної шини і її контролер, а також деякі допоміжні схеми. Типовими прикладами локальних шин є VLB і PCI.

Необхідність збереження балансу між продуктивністю і зростанням швидкодії мікропроцесорів привела до багаторівневої організації шин на основі використання декількох системних і локальних шин. Локальні шини порівняно короткі, зазвичай високошвидкісні і відповідають організацій підсистеми пам’яті для забезпечення максимальної пропускної здатності каналу пам’ять – процесор.

Шини введення-виведення (системні шини) можуть мати велику протяжність, підтримувати приєднання багатьох типів пристроїв і зазвичай слідують одному з шинних стандартів. Шина введення-виведення комп'ютера розглядається як шина розширення, що забезпечує поступове нарощування пристроїв введення-виведення. Тому стандарти відіграють величезну роль, дозволяючи розробникам комп'ютерів і пристроїв введення-виведення працювати незалежно.

Інтерфейси периферійних пристроїв (малі інтерфейси), їх призначення полягає у виконанні функцій сполучення контролера (адаптера) з конкретним механізмом ПП. Інтерфейс Universal Serial Bus (USB) призначений для підключення до комп'ютера таких зовнішніх периферійних пристроїв, як мишка, клавіатура, портативний жорсткий диск, цифрова камера, VoIP-телефон (Skype) або принтер. Теоретично, до одного host-контролера USB можна підключити до 127 пристроїв. Інтерфейс VGA призначений для підключення монітора. Інтерфейс DVI теж є інтерфейсом монітора, що розроблений, головним чином, для цифрових сигналів та для того, щоб не перетворювати цифрові сигнали графічної карти в аналогові, а потім виконувати зворотне перетворення для дисплею. Мережеві інтерфейси використовуються для LAN.

Машинні системні інтерфейси і інтерфейси введення-виведення (внутрішні інтерфейси ЕОМ) завжди розробляються паралельними або послідовно-паралельними (якщо одночасно передається не вся інформаційна одиниця, а її частина, яка містить кілька двійкових розрядів. Ці інтерфейси пов'язані багатожильним кабелем інтерфейсу (кількість жил не менше числа розрядів (бітів), які передаються одночасно). Паралельним інтерфейсом називається інтерфейс в якому забезпечується обмін одночасно всіма розрядами переданої інформаційної одиниці (найчастіше байта або машинного слова).

Використовується асинхронний і синхронний принцип обміну інформацією. У разі синхронного інтерфейсу моменти видачі інформації передаючім пристроєм і прийому її в іншому пристрої повинні синхронізуватися, для цього використовують спеціальну лінію синхронізації. При асинхронному інтерфейсі передача здійснюється за принципом «запит-відповідь». Кожен цикл передачі супроводжується послідовністю управляючих сигналів, які виробляються передавачем і прийомним пристроєм. Передавальний пристрій може здійснювати передачу даних (байти або декількох байтів) тільки після підтвердження приймачем своєю готовністю до прийому даних.

Міжмашинні інтерфейси використовуються в обчислювальних системах і мережах. Для кожного інтерфейсу характерна наявність спеціального апаратного комплексу (рис. 2.8).

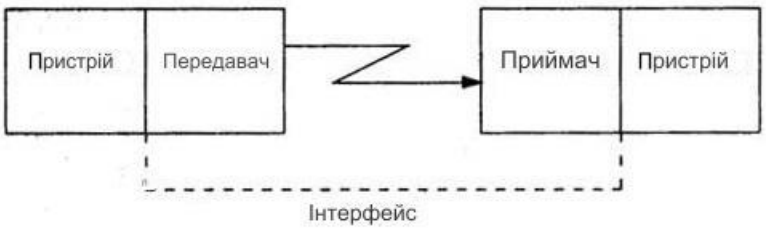

Рисунок 2.8. Місце інтерфейсу в апаратному комплексі

Характер використання цього апаратного комплексу визначається технологією обміну, способом з'єднання, ідеологією даного інтерфейсу, протоколом спілкування.

Спосіб з'єднання компонентів комплексу поділяється на радіальний, магістральний і змішаний. На рис. 2.9 представлені радіальний і магістральний інтерфейси які з'єднують центральний модуль (ЦМ) та інші модулі (компоненти) системи (М1, ..., Mn).

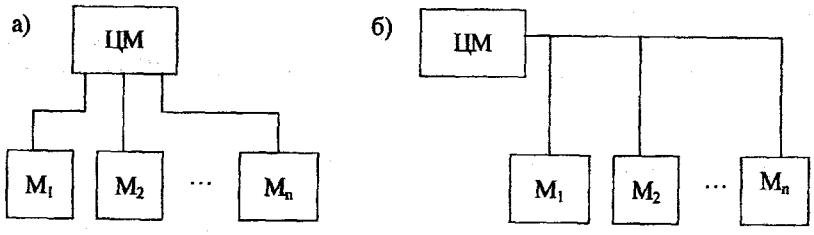

Рисунок 2.9. Радіальний (а) і магістральний (б) інтерфейси

Радіальний інтерфейс дозволяє всім модулям (М1,..., Mn) працювати незалежно, але має максимальну кількість ліній зв’язку. Магістральний інтерфейс (загальна шина) використовує принцип поділу часу для зв'язку між ЦМ і іншими модулями. Він порівняно простий в реалізації, але лімітує швидкість обміну.

Інтерфейси міжмашинного обміну зазвичай послідовні, в яких обмін інформацією здійснюється по одному біту послідовно. Для цього використовуються одно-двох провідною лінією зв’язку, світловодом, коаксіальним кабелем, радіоканалом.

Людино-машинний інтерфейс займає особливе місце, він реалізує специфічні методи організації спілкування. Наприклад, церемоніал людино-машинного спілкування передбачає "дружність користувачеві", яка може забезпечуватися технічними засобами, програмами і технологією спілкування.

Всі типи інтерфейсів характеризуються наступними параметрами:

1) пропускною спроможністю інтерфейсу - кількістю інформації яка може бути передана через інтерфейс в одиницю часу;

2) максимальної частотою передачі інформаційних сигналів через інтерфейс;

3) інформаційної шириною інтерфейсу - числом біт або байт даних, переданих паралельно через інтерфейс;

4) максимально допустимою відстанню між пристроями, які сполучаються;

5) динамічними параметрами інтерфейсу - часом передачі окремого слова або блоку даних з урахуванням тривалості процедур підготовки та завершення передачі;

6) загальним числом проводів (ліній) в інтерфейсі.

Залежно від використовуваних при обміні програмно-технічних засобів інтерфейси введення-виведення поділяються на два рівня: фізичний і логічний (рис. 2.10).

Рисунок 2.10. Логічний і фізичний рівні інтерфейсів введення-виведення

В підсистемах введення-виведення ЕОМ використовуються два способи організації передачі даних між пам’яттю і ПП: програмно-керована передача і прямий доступ до пам’яті (ПДП).

Програмно-керована передача даних здійснюється за безпосередньої участі і під управлінням процесора, який при цьому виконує спеціальну підпрограму введення-

виведення. Операція введення-виведення може ініціюватися центральним процесором, тобто поточної командою програми, або запитом переривання від ПП. Перший випадок є простим в реалізації, але при обробці команди введення-виведення ЦП марно витрачає час, коли чекає готовності ПП. Це значно знижує продуктивність ЕОМ. Програмно-керована передача, що ініціюється запитом переривання від ПП, дозволяє організувати більш гнучку взаємодію між ЦП і ПП.

Залежно від ступеня участі центрального процесора в обміні даними в інтерфейсах можуть використовуватися три способи управління обміном:

- режим сканування (так званий «асинхронний» обмін);
- синхронний обмін;
- прямий доступ до пам’яті.

Для внутрішнього інтерфейсу ЕОМ *режим сканування* передбачає опитування центральним процесором периферійного пристрою (ПП): чи готовий він до обміну, і якщо ні – продовження опитування периферійного пристрою (рис. 2.11).

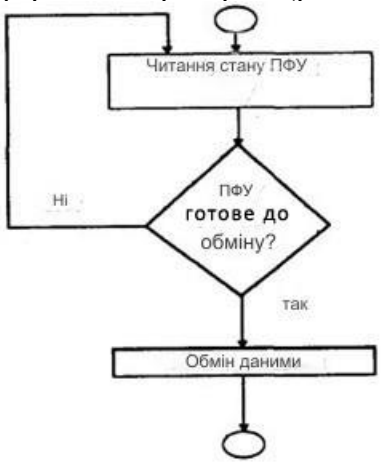

Рисунок 2.11. Алгоритм сканування

Операція пересилання даних занадто проста, щоб ефективно завантажувати складну швидкодіючу апаратуру процесора, як результат в режимі сканування знижується продуктивність обчислювальної машини. Під час пересилання блоку даних процесор для кожної одиниці переданих даних (байт, слово) виконує досить багато команд, щоб забезпечити буферізацію даних, перетворення форматів, підрахунок кількості переданих даних, формування адрес в пам’яті і т.п., як результат швидкість передачі даних при пересиланні блоку даних навіть через високопродуктивний процесор може виявитися неприйнятною для систем управління, що працюють в реальному масштабі часу.

Режим сканування спрощує підготовку до обміну, але має ряд недоліків:

- процесор постійно задіяний і не може виконувати іншу роботу;
- при великій швидкодії периферійного пристрою процесор не встигає організувати обмін даними.

У *синхронному режимі* центральний процесор виконує основну роль в організації обміну, але на відміну від режиму сканування не чекає готовності пристрою, а здійснює іншу роботу. Коли в ньому виникає потреба, зовнішній пристрій за допомогою відповідного переривання звертає на себе увагу центрального процесора.

Для швидкого введення-виведення блоків даних і розвантаження процесора від управління операціями введення-виведення використовують прямий доступ до пам’яті (DMA - Direct Memory Access).

Прямим доступом до пам’яті називається спосіб обміну даними, що забезпечує автономно від процесора встановлення зв'язку і передачу даних між основною пам’яттю і зовнішнім пристроєм.

У режимі прямого доступу до пам’яті використовується спеціалізований пристрій – контролер прямого доступу до пам’яті, який перед початком обміну програмується за допомогою центрального процесора: в нього передаються адреси основної пам’яті і кількість переданих даних. Потім центральний процесор від контролера прямого доступу до пам’яті відключається, дозволивши йому працювати, і до закінчення обміну може виконувати іншу роботу. Про закінчення обміну контролер прямого доступу до пам’яті повідомляє процесор. У цьому випадку участь центрального процесора непряма. Обмін веде контролер прямого доступу до пам’яті.

При роботі в режимі прямого доступу до пам’яті контролер ПДП послідовно виконує наступні кроки:

– приймає запит на ПДП від зовнішнього пристрою;

– формує запит мікропроцесора на захоплення шин системної магістралі;

– приймає сигнал, що підтверджує вхід мікропроцесора в стан захоплення (переходу в z-стан, при якому процесор відключається від системної магістралі);

– формує сигнал, що повідомляє зовнішній пристрій про початок виконання циклів ПДП;

– видає на шину адреси системної магістралі адресу комірки ОП, призначеної для обміну;

– виробляє сигнали, що забезпечують управління обміном даними;

– після закінчення ПДП контролер або організує повторення циклу ПДП, або припиняє режим ПДП, знімаючи запити на нього.

Цикли ПДП виконуються з послідовно розташованими комірками пам’яті, тому контролер ПДП має лічильник числа переданих байтів.

На рис. 2.15 наведена схема взаємодії пристроїв мікропроцесорної системи в режимі ПДП.

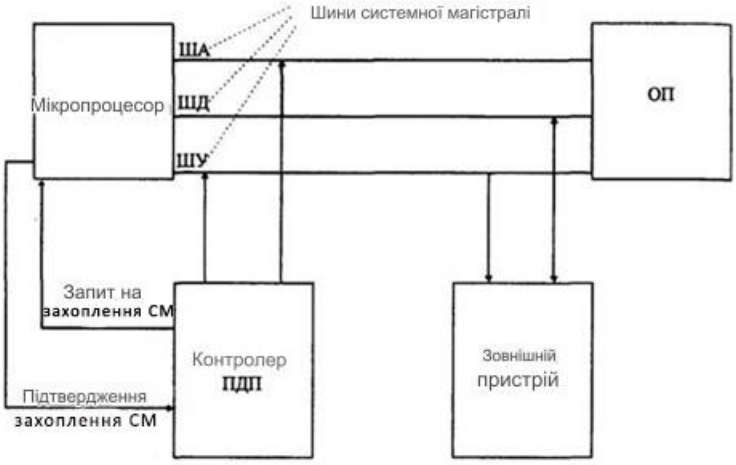

Рисунок 2.12. Взаємодія пристроїв в режимі ПДП

Контролери ПДП (КПДП) в IBM PC сумісні знизу-вгору, тобто програми, написані з використанням КПДП ранніх моделей ЕОМ, нормально працюють і на більш пізніх моделях ЕОМ.

Кожен канал КПДП складається з чотирьох 16-розрядних регістрів: регістр поточного адреси, регістр циклів ПДП, регістр зберігання базової адреси, регістр зберігання базового числа циклів ПДП і 6 розрядний регістр режиму.

Регістр поточного адреси зберігає поточну адресу комірки пам’яті при виконанні циклу ПДП. Після виконання кожного циклу ПДП вміст цього регістра збільшується або зменшується на одиницю. Він може бути прочитаний або завантажений за допомогою двох команд IN або відповідно OUT, які працюють кожна тільки з одним байтом. У режимі авто-ініціалізації вміст регістра поточної адреси при оновленні приймає базову адресу з регістра зберігання базової адреси.

Регістр циклів ПДП зберігає число слів, призначених для передачі. При виконанні циклів регістр працює в режимі віднімання лічильника. При переході з нульового стану в FFFFH виробляється керуючий сигнал для блоку управління контролером. Регістр може бути прочитаний двома командами IN. У нього можна здійснити запис двома командами OUT з мікропроцесора або в режимі ініціалізації – з регістра зберігання базового числа циклів ПДП.

Регістри BAR і WCR призначені для зберігання констант - базової адреси та базового числа циклів. Вони завантажуються в режимі програмування КПДП одночасно з регістрами CAR і CWR. В процесі виконання циклів ПДП їх вміст не змінюється. Прочитати стан цих регістрів неможливо.

Регістр режиму визначає режим роботи каналу. Він містить інформацію про номер каналу, тип циклу ПДП (читання (ОП <- ЗП), запис (ОП -> ЗП), перевірка), необхідності авто-ініціалізації, режимі зміни регістру поточного адреси (CAR) - збільшення або зменшення та режим роботи каналу - передача за запитом, одиночна передача, блокова передача, каскадування (робота у складі каскаду КПДП).

Регістр команд блоку управління режимом визначає основні параметри роботи каналу. Завантажується при програмуванні КПДП мікропроцесором.

Регістр умов зберігає дозволи на ПДП кожному каналу (що встановлюються програмно) і запам’ятовує факт переходу у регістрі зберігання базового числа циклів кожного каналу.

Контролер ПДП може працювати в двох основних режимах: в режимі програмування і режимі виконання циклів ПДП.

У режимі програмування мікропроцесор працює з КПДП, як із зовнішнім пристроєм. Після завантаження в КПДП керуючих слів контролер переходить в пасивний стан. У цьому стані КПДП буде перебувати до тих пір, поки не надійде запит на ПДП від ЗП або від мікропроцесора. Виявивши запит на ПДП, контролер виставляє у мікропроцесор запит на захоплення системної магістралі і очікує від нього підтвердження захоплення (тобто відключення МП від СМ, переходу його в стан високого опору, Z-стану). При отриманні сигналу підтвердження захоплення, контролер починає виконувати цикли ПДП.

Системна магістраль (СМ) в режимі прямого доступу до пам’яті використовується мультиплексно КПДП і мікропроцесором, причому головне управління системною магістраллю виконує КПДП, а МП отримує до неї доступ на дуже короткі проміжки часу для обміну інформацією з основною пам’яттю. Незважаючи на велику самостійність ПДП, провідне становище в МП – системі залишається за мікропроцесором, який, запустивши обмін інформацією в режимі ПДП, продовжує виконувати свою роботу.

Системна магістраль або шина – це середовище передачі сигналів управління, адрес, даних, до якої паралельно і одночасно може підключатися кілька компонентів обчислювальної системи. Фізично системна магістраль є паралельні провідники на материнській платі, які називаються лініями, алгоритми, по яких передаються сигнали, правила інтерпретації сигналів, дисципліни обслуговування запитів, спеціальні мікросхеми, що забезпечують цю роботу. Весь цей комплекс утворює поняття інтерфейс системної магістралі або стандарт обміну. Системна магістраль є вузьким місцем ЕОМ, так як всі пристрої, підключені до неї, конкурують за можливість передавати свої дані по її шинам.

Шини можна розділити на групи відповідно до виконуваних функцій. Вони можуть бути внутрішніми по відношенню до процесора і служити для передачі даних в АЛУ і з АЛУ, а можуть бути зовнішніми по відношенню до процесора і пов’язувати процесор з пам’яттю або пристроями введення-виведення. Кожен тип шини має певні властивості, і до кожного з них пред’являються певні вимоги.

Сучасні персональні комп’ютери зазвичай містять спеціальну шину між центральним процесором і пам’яттю і принаймні ще одну шину для пристроїв введення-виведення. На рис. 7 зображена система з однією шиною пам’яті і однієї шиною введення-виведення.

У літературі шини зазвичай зображуються у вигляді жирних стрілок. Коли всі біти одного типу, наприклад адресні або інформаційні, малюється звичайна стрілка. Коли

включаються адресні лінії, лінії даних і управління, використовується жирна стрілка (рис. 2.13).

Робота шини і всіх пристроїв, які пов’язані з шиною, повинні відповідати правилам, які називаються протоколом шини. Крім того, існують певні технічні вимоги, щоб плати від третіх виробників підходили до каркасу для друкованих плат і мали роз’єми, відповідні материнській платі механічно і з точки зору потужностей, синхронізації і т. п.

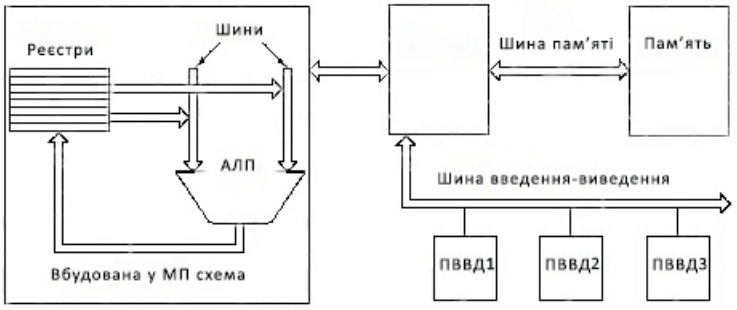

Рисунок 2.13. Комп’ютерна система з декількома шинами

Існує ряд широко використовуваних в комп'ютерному світі шин: Omnibus (PDP-8), Unibus (PDP-11), IBM PC (PC / XT), ISA (PC / AT), EISA (80386), MicroChannel (PC / 2), PCI (різні персональні комп'ютери), SCSI (різні персональні комп’ютери і робочі станції), Nubus (Macintosh), Universal Serial Bus (сучасні персональні комп’ютери), FireWire (побутова електроніка).

Деякі пристрої, пов’язані з шиною, є активними і можуть ініціювати передачу інформації по шині, тоді як інші є пасивними і чекають запитів. Активний пристрій називається пристроєм, що задає дію, пасивний - підлеглим пристроєм.

Коли центральний процесор вимагає від контролера диска зчитати або записати блок інформації, центральний процесор діє як пристрій, що задає дію, а контролер диска - як підлеглий пристрій. Контролер диска може діяти як пристрій, що задає дію, коли він командує пам’яті прийняти слова, які зчитує з диска. Кілька типових комбінацій пристрою, що задає дію і підлеглого пристрою вказані в табл. 1. Пам’ять ні за яких обставин не може бути пристроєм, що задає.

Таблиця 2.1. Приклади пристроїв, що задають дію і підлеглих пристроїв

Пристрій, що задає дію	Підлегле пристрій	Приклад
Центральний процесор	Пам’ять	Виклик команд і даних
Центральний процесор	Пристрій введення-виведення	Ініціалізація передачі даних
Центральний процесор	Співпроцесор	Передача команди від процесора до співпроцесору
Пристрій введення-виведення	Пам’ять	ПДП (прямий доступ до пам’яті)
Співпроцесор	Центральний процесор	Виклик співпроцесором операндів з центрального процесора

Двійкові сигнали, які видають пристрої комп’ютера, часто недостатньо інтенсивні, щоб активізувати шину, особливо якщо вона досить довга і якщо до неї приєднано багато пристроїв. З цієї причини більшість пристроїв, що задають дію зазвичай пов’язані з шиною через мікросхему, яка називається драйвером шини, по суті є двійковим підсилювачем.

Подібним чином більшість підлеглих пристроїв пов’язані з шиною приймачем шини. Для пристроїв, які можуть як задавати дію, так бути і підлеглим пристроєм, використовується приймач шини. Ці мікросхеми взаємодії з шиною часто є пристроями з трьома станами, що дає їм можливість від’єднуватися, коли вони не потрібні. Іноді вони підключаються через відкритий колектор, що дає подібний ефект. Коли один або декілька

пристроїв на відкритому колекторі вимагають доступу до шини в один і той же час, результатом є булева операція АБО над усіма цими сигналами. Така угода називається монтажним АБО. У більшості шин одні лінії є пристроями з трьома станами, а інші, яким потрібна властивість монтажного АБО, - відкритим колектором.

Як і процесор, шина має адресні лінії, інформаційні лінії і лінії управління. Проте між командами процесора і сигналами шини може і не бути взаємно однозначної відповідності. Наприклад, деякі процесори містять команди, які видають сигнал читання з пам'яті або запису в пам'ять, або читання пристрої введення-виведення, або запису на пристрій введення-виведення, або будь-які інші операції. Звичайна шина може містити одну лінію для читання з пам'яті, другу лінію для запису в пам'ять, третю – для читання пристрою введення-виведення, четверту – для запису на пристрій введення-виведення і т. п. Мікросхема-декодер повинна пов'язувати даний процесор з такою шиною, щоб перетворювати 3-бітний кодований сигнал в окремі сигнали, які можуть управляти лініями шини.

Ширина шини пов'язана з кількістю адресних ліній. Чим більше адресних ліній містить шина, тим до більшого обсягу пам'яті може звертатися процесор. Якщо шина містить n адресних ліній, тоді процесор може використовувати її для звернення до 2n різних комірок пам'яті. Для пам'яті великої ємності де необхідно багато адресних ліній, використовується мультиплексна шина. У цієї шини немає поділу на адресні та інформаційні лінії. У ній може бути, наприклад, 32 лінії і для адрес, і для даних. Спочатку ці лінії використовуються для адрес. Потім вони використовуються для даних. Щоб записати інформацію в пам'ять, потрібно спочатку передавати в пам'ять адресу, а потім дані. У випадку з окремими лініями адреси і дані можуть передаватися разом. Об'єднання ліній скорочує ширину і вартість шини, але система працює при цьому повільніше.

Шини можна розділити на дві категорії в залежності від їх синхронізації. Синхронна шина містить лінію, яка запускається кварцовим генератором. Сигнал на цій лінії є меандр з частотою зазвичай від 5 до 100 МГц. Будь-яка дія шини займає ціле число так званих циклів шини. Асинхронна шина не містить, генератора. Цикли шини можуть бути будь-якої необхідної довжини і необов'язково однакові по відношенню до всіх пар пристроїв.

Мікросхеми введення-виведення можуть ставати пристроєм, що задає дію при зчитуванні інформації з пам'яті і запису інформації в пам'ять. Крім того, вони можуть викликати переривання. Співпроцесори також можуть ставати пристроєм шини, що задає дію. Щоб запобігти хаосу, який може статися під час звернення до шини кількох пристроїв одночасно, потрібен спеціальний механізм – так званий арбітраж шини.

Механізми арбітражу може визначати тільки наявність або відсутність запитів. Коли арбітр бачить запит шини, він запускає лінію надання шини. Ця лінія послідовно пов'язує всі пристрої введення-виведення (як в ялинкової гірлянди). Коли фізично найближчий до арбітра пристрій сприймає сигнал надання шини, арбітр перевіряє, чи немає запиту шини. Якщо запит є, пристрій користується шиною, але не поширює сигнал надання далі по лінії. Якщо запиту немає, пристрій передає сигнал надання шини наступному пристрою. Цей пристрій теж перевіряє, чи є запит, і діє відповідним чином залежно від наявності або відсутності запиту. Передача сигналу надання шини триває до тих пір, поки який-небудь пристрій не скористається наданою шиною. Така система називається системою послідовного опитування. При цьому пріоритети пристроїв залежать від того, наскільки близько вони знаходяться до арбітра. Щоб обійти таку систему, в якій пріоритети залежать від відстані до арбітра, в деяких шинах діє кілька рівнів пріоритету. На кожному рівні пріоритету є лінія запиту шини і лінія надання шини.

Деякі арбітри містять третю лінію, яка запускається, як тільки пристрій приймає сигнал надання шини, і бере шину в своє розпорядження. Як тільки запускається ця лінія підтвердження прийому, лінії запиту і надання шини можуть бути відключені. В результаті інші пристрої можуть запитувати шину, поки перший пристрій використовує її. До того

моменту, коли закінчиться поточна передача, наступний пристрій, що задає дію вже буде обрано. Цей пристрій може почати роботу, як тільки відключається лінія.

Зазвичай за раз передається одне слово. При використанні кеш-пам'яті бажано відразу викликати весь рядок кеш-пам'яті (тобто 16 послідовних 32-бітних слів). Часто передача блоками може бути більш ефективна, ніж така послідовна передача інформації. Коли починається читання блоку, пристрій повідомляє підлеглому пристрою, скільки слів потрібно передати (наприклад, поміщаючи загальне число слів на інформаційні лінії в період T_i).

Ще один важливий цикл шини – цикл для здійснення переривань. Коли центральний процесор командує пристрою введення-виведення зробити якусь дію, він очікує переривання після завершення роботи. Для сигналу переривання потрібна шина.

Оскільки може скластися ситуація, коли кілька пристроїв одночасно хочуть зробити переривання, тут мають місце ті ж проблеми вирішення конфліктних ситуацій, що і в звичайних циклах шини. Щоб уникнути таких проблем, потрібно кожному пристрою приписати певний пріоритет і використовувати централізований арбітр для розподілу пріоритетів.

Перериванням програм називається процес реакції ЕОМ на події, які можуть виникати під час виконання поточної програми всередині машини і у зовнішньому середовищі (технологічний процес, керований ЕОМ), пов'язаному з нею.

Реакція полягає в тому, що машина перериває обробку поточної програми і переходить до виконання деякої іншої програми, спеціально призначеної для даної події. По завершенню цієї програми ЕОМ повертається до виконання перерваної програми. Розглянутий процес, називається (рис. 2.14).

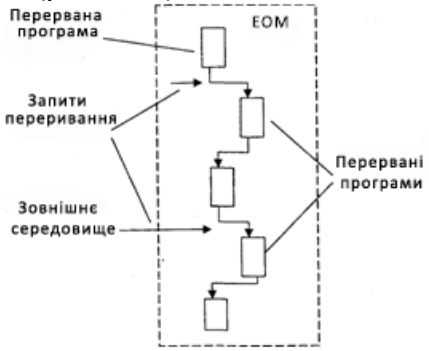

Рисунок 2.14. Процес переривання програми

Принципово важливим є те, що моменти виникнення подій, що вимагають переривання програм, заздалегідь не відомий і тому не може бути врахований під час програмування.

Кожна подія, що вимагає переривання, супроводжується сигналом, який називають запитом переривання.

Програму, затребувану запитом переривання, називають програмою, що ініціює переривання, протиставляючи їй програму, яка переривається під час виконання у ЕОМ до появи запиту.

Запити на переривання можуть виникати всередині самої ЕОМ і в її зовнішньому середовищі. До перших відносяться, наприклад, запити при виникненні в ЕОМ таких подій, як поява помилки в роботі її апаратури, переповнення розрядної сітки, спроба поділу на 0, вихід з області пам'яті, який відповідає даній програмі, операції введення-виведення, що затребував периферійний пристрій, завершення операції введення-виведення периферійним пристроєм або виникнення при цій операції особливої ситуації та ін. Хоча деякі із зазначених подій породжуються самою програмою, моменти їх появи, як правило, неможливо передбачити. Запити в зовнішньому середовищі можуть виникати від інших ЕОМ, від аварійних та деяких інших датчиків технологічного процесу і т.п.

Таким чином, запити переривання генеруються декількома процесами, що розвиваються паралельно в часі, які в деякі моменти вимагають втручання процесора.

До цих процесів, зокрема, відноситься процес виконання самої програми, процес контролю правильності роботи ЕОМ, операції введення-виведення, технологічний процес в керованому машиною об’єкті та ін.

Можливість переривання програм – важлива архітектурна властивість ЕОМ, що дозволяє ефективно використовувати продуктивність процесора при наявності декількох процесів, що протікають паралельно в часі і вимагають в довільні моменти часу обслуговування і управління з боку процесора. В першу чергу це відноситься до організації паралельної у часі роботи процесора і периферійних пристроїв машини, а також до використання ЕОМ для управління в реальному часі технологічними процесами.

Для реалізації переривання програм з високою швидкодією, існують відповідні апаратурні і програмні засоби, сукупність яких називається *системи переривання* програм. Як апаратний засобів використовується *контролер переривання* (блок переривання).

Основними функціями системи переривання є:

– запам’ятовування стану програми, що переривається і здійснення переходу до програми, яка перериває;

– відновлення стану перерваної програми і повернення до неї. При наявності декількох джерел запитів переривання між ними повинні бути встановлені пріоритетні співвідношення, що визначають, який з запитів, що надійшли, підлягає обробці в першу чергу, і встановлюють чи має право чи ні даний запит (програма, що перериває) переривати ту чи іншу програму.

Характеристики системи переривання

Для оцінки ефективності систем переривання можуть бути використані наступні характеристики.

1. *Загальна кількість запитів переривання* (входів в систему переривання).
2. *Час реакції* - час між появою запиту переривання і моментом переривання поточної програми. На рис. 2.15 наведена спрощена тимчасова діаграма процесу переривання.

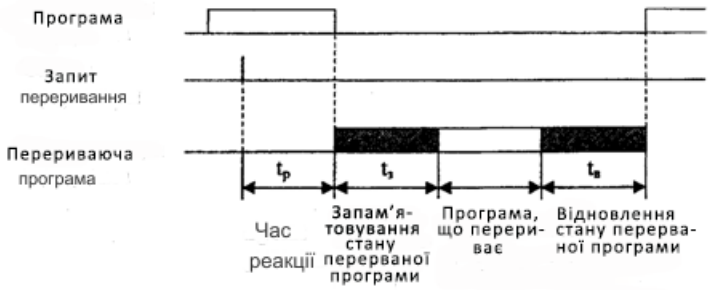

Рисунок 2.15. Спрощена тимчасова діаграма процесу переривання

Для одного і того ж запиту затримки у виконанні програми, що перериває, залежать від того, скільки програм зі старшим пріоритетом чекають обслуговування, тому час реакції визначають для запиту з найвищим пріоритетом (tp).

Час реакції залежить від того, в який момент допустимо переривання. Здебільшого переривання допускається після закінчення поточної команди. В цьому випадку час реакції визначається в основному тривалістю виконання команди.

Цей час реакції може виявитися неприпустимо великим для ЕОМ, призначених для роботи в реальному масштабі часу. У таких машинах допускається переривання після будь-якого такту виконання команди (мікрокоманд). Однак при цьому зростає кількість інформації, що підлягає запам’ятовуванню і відновленню при перемиканні програм, так як в цьому випадку необхідно зберігати також і стан лічильника тактів в момент переривання,

регістра коду операції та деяких інших вузлів, тому така організація переривання можлива тільки в машинах з швидкодіючою надоперативною пам'яттю.

Є ситуації, в яких бажано негайне переривання. Якщо апаратура контролю виявила помилку, то доцільно відразу ж перервати операцію, поки помилка не зробила вплив на наступні такти роботи програми.

3. *Витрати часу на перемикання програм* (витрати переривання) дорівнюють сумарним витратам часу на запам'ятовування і відновлення стану програми (рис. 2.15).

4. *Глибина переривання* – максимальне число програм, які можуть переривати одна одну. Якщо після переходу до програми, яка перериває і аж до її закінчення прийом запитів припиняється, то кажуть, що система має глибину переривання, яка дорівнює 1. Глибина дорівнює n, якщо допускається послідовне переривання до n програм. Глибина переривання зазвичай збігається з числом рівнів пріоритету в системі переривання. На рис. 2.16 показані процеси переривання в системах з різною глибиною переривання (передбачається, що пріоритет кожного наступного запиту вище попереднього). Система з великим значенням глибини переривання забезпечує швидшу реакцію на термінові запити.

Якщо запит виявиться не обслуженим до моменту приходу нового запиту від того ж джерела, то виникає так зване насичення системи переривання. У цьому випадку попередній запит від даного джерела буде загублений, що неприпустимо.

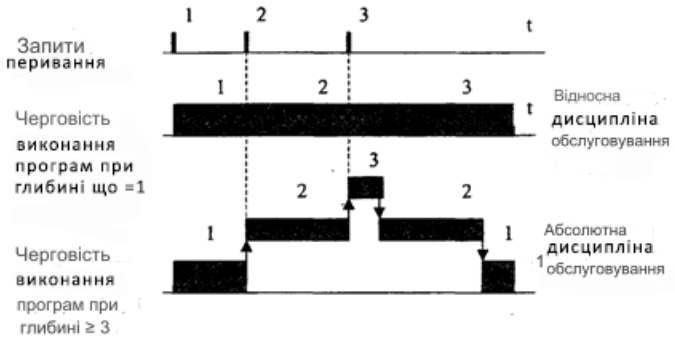

Рисунок 2.16. Процеси переривання з різною глибиною переривання і дисципліною обслуговування

5. *Число класів (рівнів) переривання.* У ЕОМ число різних запитів (причин) переривання може досягати декількох десятків або сотень. У таких випадках частина запитів поділяють на окремі класи або рівні.

Сукупність запитів, що ініціюють одну і ту ж програму, що переривається, утворює клас або рівень переривання. Запити всіх джерел переривання надходять на регістр запитів переривання, встановлюючи відповідні його розряди в стан 1, яке вказує на наявність запиту переривання певного джерела.

Інформація про дійсну причину переривання, що породила запит даного класу, міститься в коді переривання, який відображає стан розрядів регістр запитів.

Об'єднання запитів в класи переривання дозволяє зменшити обсяг апаратури, але призводить до уповільнення роботи системи переривання.

Програмно-керований пріоритет програм, що переривають

Відносна ступінь важливості програм, їх частота повторення, відносна ступінь терміновості в ході обчислювального процесу можуть змінюватися, вимагаючи встановлення нових пріоритетних відносин. Тому в багатьох випадках пріоритет між програмами, що переривають не може бути зафіксований раз і назавжди. Необхідно мати можливість змінювати в міру необхідності пріоритетні співвідношення програмним шляхом. Пріоритет між програмами, що переривають повинен бути динамічним, тобто програмно керованим.

У ЕОМ широко застосовуються два способи програмно-керованого пріоритету програм, що переривають:

– використання порога переривання;

– використання маски переривання.

Поріг переривання – цей спосіб дозволяє задавати поріг переривання, т. е. мінімальний рівень пріоритету запитів, яким дозволяється переривати програму, що виконує процесор.

Поріг переривання задається командою програми, яка встановлює в регістрі порога переривання код порога переривання. Спеціальна схема виділяє найбільш пріоритетний запит, порівнює його пріоритет з порогом переривання і, якщо він буде вище ніж поріг, виробляє загальний сигнал переривання, і починається процедура переривання.

Маска переривання це двійковий код, розряди якого поставлені у відповідність запитам або класами (рівням) переривань. Маска завантажується командою програми в регістр маски. Таким чином, програма, змінюючи маску в регістрі маски, може встановлювати довільні пріоритетні співвідношення між програмами без повторного перемикання ліній, за якими надходять запити переривання. Кожна програма, що перериває може встановити свою маску.

Замаскований запит, в залежності від причини переривання, або ігнорується, або запам’ятовується, з тим щоб виконати необхідні дії, коли заборона буде знята. Наприклад, якщо переривання викликано закінченням операції в ОП (оброблювач переривань), то його слід, як правило, запам’ятати.

Переривання, викликане переповненням розрядної сітки при арифметичній операції, слід під час його маскування ігнорувати, тому що запам’ятовування цього запиту може вплинути на частину програми або іншу програму, до яких це переповнення не відноситься.

Вектор початкового стану програми, що перериває називають вектором переривання. Він містить всю необхідну інформацію для переходу до програми, що перериває, в тому числі її початкову адресу. Кожному запиту (рівню) переривання відповідає свій вектор переривання, здатний ініціювати виконання відповідної програми, що перериває. Вектори переривання зазвичай знаходяться в спеціально виділених фіксованих осередках пам’яті (стек).

Найбільш гнучким і динамічним є векторне переривання, при якому джерело переривання, виставляючи запит переривання, посилає в процесор (виставляє на шини інтерфейсу) код адреси в пам’яті свого вектора переривання.

При векторному перериванні кожному запиту переривання відповідає перехід до початкової адреси відповідної програми, що перериває, що задається вектором переривання.

2.8. Багаторівнева структура ЕОМ

Для задоволення запитів користувачів, враховуючи фізичну структуру і можливості ЕОМ використовується багаторівнева структура. Кількість рівнів для сучасних машини доходить до 6-ти (рис. 2.17).

На самому нижньому, цифровому логічному рівні, об’єкти називаються вентилями. Хоча вентилі складаються з аналогових компонентів, таких як транзистори, вони можуть бути точно змодельовані як цифрові засоби. У кожного вентиля є одне або кілька цифрових вхідних даних (сигналів, що представляють 0 або 1). Вентилі обчислюють прості функції цих сигналів, такі як І або АБО. Кожен вентиль формується з декількох транзисторів. Кілька вентилів формують 1 біт пам’яті, який може містити 0 або 1. Біти пам’яті об’єднуються у групи, наприклад, по 16, 32 або 64, формують регістри. Кожен регістр може містити 1 двійкове число до певної межі. З вентилів може складатися сам комп’ютер.

Мікроархітектурний рівень – містить сукупності з 8-ми або 32-х регістрів, які формують локальну пам’ять і схему АЛП. Завдання цього рівня інтерпретація команд рівня архітектури команд. Загальних принципів розробки мікроархітектурного рівня не існує. Кожна розробка індивідуальна. Мікроархітектура містить мікропрограму, яка запускає окремі вентилі апаратного забезпечення.

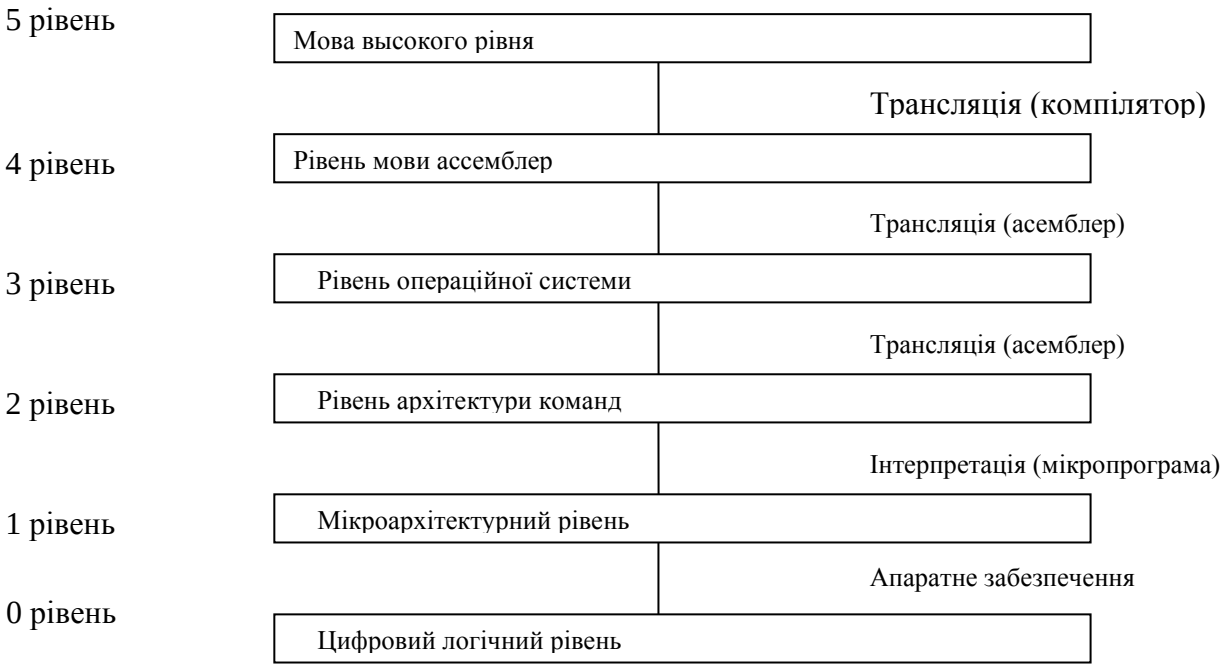

Рисунок 2.17. Багаторівнева структура ЕОМ

Рівень архітектури команд розташований між мікроархітектурним рівнем і рівнем операційної системи. Є сполучною ланкою між програмним і апаратним забезпеченням. Пов'язує компілятори і апаратне забезпечення. Це мова, яка зрозуміла і компіляторам, і апаратному забезпеченню. На рівні команд є два режими: привілейований – призначений для запуску операційної системи і призначений для користувача – для запуску програмних додатків. Команди на цьому рівні складаються з двох частин – опису операції і індексації.

2.9. Цифровий логічний рівень

Основні елементи, з яких конструюються цифрові комп'ютери, надзвичайно прості. При конструюванні цих елементів використовується спеціальна двозначна алгебра (булева алгебра).

Цифрові схеми можуть конструюватися з невеликого числа простих елементів шляхом поєднання цих елементів в різних комбінаціях.

Цифрова схема – це схема, в якій є тільки два логічних значення. Зазвичай сигнал від 0 до 1В представляє одне значення (наприклад, 0), а сигнал від 2 до 5В – інше значення (наприклад, 1). Напруга за межами зазначених величин неприпустима. Крихітні електронні пристрої, які називаються вентилями, можуть обчислювати різні функції від цих двозначних сигналів. Ці вентилі формують основу апаратного забезпечення, на якій будуються всі цифрові комп'ютери.

Розглянемо основний принцип. Вся сучасна цифрова логіка ґрунтується на тому, що транзистор може працювати як дуже швидкий бінарний перемикач. На рисунку 2.18(а) зображений біполярний транзистор, вбудований в просту схему. Транзистор має три з'єднання із зовнішнім світом; колектор, базу і емітер. Якщо вхідна напруга V_{in} нижче певного критичного значення, транзистор вимикається і діє як дуже великий опір. Це призводить до вихідному сигналу V_{out}, близькому до V_{cc} (напрузі, що подається ззовні), зазвичай +5 В для даного типу транзистора. Якщо V_{in}, перевищує критичне значення, транзистор вмикається і діє як провідник, викликаючи заземлення сигналу V_{out} (за згодою 0В).

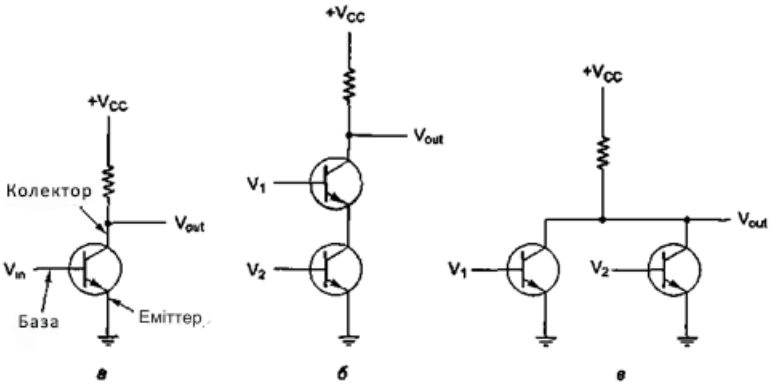

Рисунок 2.18. Транзисторний інвертор (а); вентиль НЕ-І (б); вентиль НЕ-АБО (в)

Важливо відзначити, що якщо напруга V_{in} низька, то V_{out} висока, і навпаки. Ця схема, таким чином, є інвертором, що перетворює логічний 0 в логічну 1 і логічну 1 в логічний 0. Резистор (ламана лінія) потрібен для обмеження кількість струму, що проходить через транзистор, щоб транзистор не згорів. На перемикання з одного стану до іншого зазвичай потрібно кілька наносекунд.

На рис. 2.18(б) два транзистора з'єднані послідовно. Якщо і напруга V_1 і напруга V_2 високі, то обидва транзистора будуть служити провідниками і знижувати V_{out}. Якщо одна з вхідних напруг низька, то відповідний транзистор буде вимикатися, і напруга на виході буде високою. Іншими словами, V_{out} буде низьким тоді і тільки тоді, коли і напруга V_1, і напруга V_2 високі.

На рис. 2.18(в) два транзистора з'єднані паралельно. Якщо один з вхідних сигналів високий, буде включатися відповідний транзистор і знижувати вихідний сигнал. Якщо обидві напруги на вході низькі, то вихідна напруга буде високою.

Ці три схеми утворюють три найпростіших вентиля. Вони називаються вентилями НЕ, НЕ-І та НЕ-АБО. Вентилі НЕ часто називають інверторами. Якщо прийняти угоду, що висока напруга (V_{cc}) - це логічна 1, а низька напруга («земля») – логічний 0, то можемо виражати значення на виході як функцію від вхідних значень. Значки, які використовуються для зображення цих трьох типів вентилів, показані на рис. 2.19(а)-(в) з поведінкою функції для кожної схеми. На цих рисунках А і В - це вхідні сигнали, а Х - вихідний сигнал. Кожен рядок таблиці визначає вихідний сигнал для різних комбінацій вхідних сигналів.

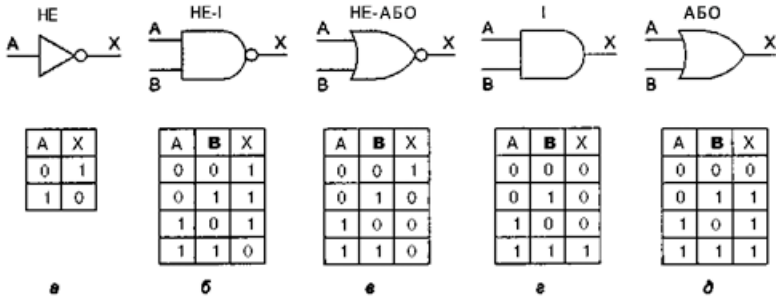

Рисунок 2.19. Значки для зображення 5 основних вентилів. Поведінка функції для кожного вентиля

Якщо вихідний сигнал (рис. 2.19(б)) подати в інвертор, ми отримаємо іншу схему, протилежну вентиля НЕ-І, тобто таку схему, у якій вихідний сигнал дорівнює 1 тоді і тільки тоді, коли обидва вхідних сигнали дорівнюють 1. Така схема називається вентилем І, її схематичне зображення і опис відповідної функції представлені на рисунку 2.19(г). Таким же чином вентиль НЕ-АБО може бути пов'язаний з інвертором. Тоді вийде схема, у якій

вихідний сигнал дорівнює 1 в тому випадку, якщо хоча б один з вхідних сигналів 1, і дорівнює 0, якщо обидва вхідні сигнали дорівнюють 0. Зображення цієї схеми, яка називається вентилем АБО, а також опис відповідної функції представлені на рисунку 2.19(д). Маленькі кружечки в схемах інвертора, вентиля НЕ-І та вентиля НЕ-АБО називаються виходами, що інвертуються. Вони також можуть використовуватися в іншому контексті для вказівки на інвертований сигнал.

П’ять вентилів, зображених на рисунку 2.19, складають основу цифрового логічного рівня. З попереднього обговорення було з’ясовано, що вентилі НЕ-І та НЕ-АБО потребують два транзистора кожен, а вентилі І і АБО - три транзистора кожен. З цієї причини в багатьох комп’ютерах використовуються вентилі НЕ-І та НЕ-АБО, а не І і АБО. (На практиці всі вентилі виконуються дещо по-іншому, але НЕ-І та НЕ-АБО все одно простіше, ніж І і АБО.) Варто згадати, що вентилі можуть мати більше двох входів. В принципі вентиль НЕ-І, наприклад, може мати довільну кількість входів, але на практиці більше восьми зазвичай не буває.

Хоча будова вентилів відноситься до рівня фізичних пристроїв, згадаємо основні виробничі технології, так як вони згадуються в літературі. Дві основні технології - біполярна і МОН (метал-оксид-напівпровідник). Серед біполярних технологій можна назвати ТТЛ (транзисторно-транзисторну логіку), яка служила основою цифрової електроніки протягом багатьох років, і ЕСЛ (еміттерну – пов’язану логіку), яка використовується в тих випадках, коли потрібна висока швидкість виконання операцій.

Вентилі МОН працюють повільніше, ніж ТТЛ і ЕСЛ, але споживають набагато менше енергії і займають набагато менше місця, тому можна компактно розмістити велику кількість таких вентилів. Вентилі МОН мають кілька різновидів: p-канальний МОН-прилад, n-канальний МОН-прилад і комплементарний МОН. Хоча МОН-транзистори конструюються не так, як біполярні транзистори, вони мають таку саму здатність функціонувати, як електронні перемикачі. Сучасні процесори і пам’ять найчастіше виробляються з використанням технології компліментарних МОН, яка працює при напрузі +3,3В.

Інтегральні схеми

Вентилі виробляються в модулях і називаються інтегральними схемами (ІС) або мікросхемами. Мікросхеми діляться на класи в залежності від кількості вентилів.

МІС (мала): 1-10 вентилів.

СІС (середня): 1-100.

БІС (велика): 100 -100000.

НБІС (надвелика): більше 100000 (до 10 млн).

Комбінаційні схеми

Схема, в якій вихідні сигнали визначаються поточними вхідними сигналами, називається комбінаційної схемою. Наприклад, схема, яка містить елементи пам’яті, може генерувати вихідні сигнали, які залежать від значень, що зберігаються в пам’яті.

Мультиплексор: є схемою з 2*n* входами, одним виходом і *n* лініями управління, які вибирають один з входів. Обраний вхід з’єднується з виходом. Використовується для реалізації «таблиці істинності», а також в якості перетворювача паралельного коду в послідовний. Таке перетворення здійснюється при введенні інформації з клавіатури.

Декодери: схема, яка отримує на вході *n*-бітове число і використовує його для того, щоб вибрати (тобто встановити значення 1) одну з 2*n* вихідних ліній. Використовується для запуску однієї мікросхеми пам’яті з 8-ми.

Компаратори: приймає два вхідних сигнали, кожен довжиною 4 біта і видає 1, якщо вони рівні і 0, якщо не рівні. Заснована на вентилі «НЕ-АБО». Використовується для порівняння двох слів що надходять на вхід.

Програмовані логічні матриці: мікросхема містить входи для 12 змінних. Додаткові сигнали (інверсії) генеруються всередині самої мікросхеми. У підсумку виходить 24 вхідних сигналів. Використовується для обчислення суми добутків.

Схема зсуву: містить рівну кількість входів і виходів, наприклад 8. Використовується для зсуву вхідних сигналів на 1 біт, лінія управління визначає напрямок зсуву: 0 – наліво, 1 – направо. Була першою арифметичної схемою.

Суматори: схема виконує додавання і перенесення на наступну позицію. Використовується для виконання операцій додавання.

Арифметико-логічний пристрій: схема для *n*-бітових слів, яка складається з *n* ідентичних схем для індивідуальних бітових позицій. Використовується для виконання операцій І, АБО і складання над двома машинними словами.

Тактові генератори: схема, яка викликає серію імпульсів. Всі імпульси однакові за тривалістю. Інтервали між послідовними імпульсами також однакові. Часовий інтервал між початком одного імпульсу і початком наступного називається часом такту. Вбудовується в цифрові схеми для контролю тимчасових відносин, щоб забезпечити синхронізацію (порядок виконання дій).

Схеми пам'яті

Засувки потрібні щоб створити один біт пам'яті, така схема «запам'ятовує» попередні вхідні значення. Цю схему можна сконструювати з двох вентилів НЕ-АБО. Аналогічні схеми можна побудувати з вентилів НЕ-І.

Тригери вибирають значення на певній лінії в певний момент часу і запам'ятовують його. У такій схемі перехід стану відбувається не тоді, коли сигнал, що синхронізує дорівнює 1, а під час переходу сигналу, що синхронізує з 0 на 1 (наростаючий фронт) або з 1 на 0 (задній фронт). Отже, довжина імпульсу, що синхронізує не має значення, оскільки переходи відбуваються швидко.

Тригер запускається фронтом сигналу, а *засувка* запускається рівнем сигналу.

Регістри представляють собою об'єднання ліній синхронізації і ліній очищення в одній мікросхемі.

2.10. Мікроархітектурний рівень

Над цифровим логічним рівнем знаходиться мікроархітектурний рівень. Його задача - інтерпретація команд рівня архітектури команд. Побудова мікроархітектурного рівня залежить від того, який рівень архітектури команд, а також від вартості і призначення комп'ютера.

Рівень архітектури команд часто містить прості команди, які виконуються за один цикл.

В інших системах на цьому рівні є більш складні команди; виконання однієї такої команди займає кілька циклів.

Щоб виконати команду, потрібно знайти операнди в пам'яті, зчитати їх і записати отримані результати в пам'ять. Управління рівнем команд зі складними командами відрізняється від управління рівнем команд з простими командами, так як в першому випадку виконання однієї команди вимагає певної послідовності операцій.

На жаль, таких загальних принципів розробки мікроархітектурного рівня не існує. Кожна розробка індивідуальна. З цієї причини просто докладно розглянемо конкретний приклад, підмножину віртуальної машини Java. Це підмножина містить тільки команди з цілими числами. Система містить кілька досить складних команд. Подібні архітектури часто реалізуються за допомогою мікропрограмування.

Мікроархітектура містить мікропрограму (в ПЗУ), яка повинна викликати, декодувати і виконувати команди, це крихітна мікропрограма, яка запускає окремі вентилі апаратного забезпечення. Реальне апаратне забезпечення складається тільки з компонентів.

Розробку цієї мікроархітектури зручно вважати проблемою програмування, при цьому кожна команда рівня архітектури команд – функція, що викликається основною програмою. В даному випадку основна програма досить проста. Вона являє собою нескінченний цикл.

Спочатку програма визначає, яку функцію потрібно виконати, потім викликає цю функцію, а потім починає все знову.

Мікропрограма містить набір змінних, до яких мають доступ усі функції. Цей набір змінних називається станом комп'ютера. Кожна функція змінює принаймні кілька змінних, формуючи при цьому стан. Наприклад, лічильник команд – це частина стану. Він вказує місцезнаходження функції (тобто команди рівня архітектури команд), яку потрібно виконати наступною. Під час виконання кожної команди лічильник команд вказує на наступну команду.

Команди дуже короткі. Кожна команда складається з декількох полів, зазвичай одного або двох, кожне з яких виконує певну функцію.

Перше поле є кодом операції. Цей код визначає тип команди і повідомляє, що це, наприклад, команда додавання або команда розгалуження, або ще якась команда. Багато команд містять додаткове поле, яке визначає тип операнда. Наприклад, команди, які мають доступ до локальних змінних, повинні мати спеціальне поле, щоб визначити, яка це змінна.

Така модель виконання команди, яка називається іноді циклом вибірка-виконання і може служити основою втілення рівня архітектури команд зі складними командами.

Повний список команд формує мікропрограму.

2.11. Рівень архітектури команд

Рівень архітектури команд має особливе значення: він є сполучною ланкою між програмним і апаратним забезпеченням. Звичайно, можна було б зробити так, щоб апаратне забезпечення відразу безпосередньо виконувало програми, написані на C, C ++, FORTRAN 90 або іншими мовами високого рівня, але це не дуже гарна ідея. Перевага компіляції перед інтерпретацією було б тоді втрачено. Крім того, з чисто практичних міркувань комп'ютери повинні вміти виконувати програми, написані на різних мовах.

По суті, всі розробники вважають, що потрібно транслювати програми, написані на різних мовах високого рівня, в загальну проміжну форму - на рівень архітектури команд - і відповідно конструювати апаратне забезпечення, яке може безпосередньо виконувати програми цього рівня (рівня архітектури команд). Рівень архітектури команд пов'язує компілятори і апаратне забезпечення. Це мова, яка зрозуміла і компіляторам, і апаратному забезпеченню.

В ідеалі при створенні нової машини розробники архітектури команд повинні консультуватися і з укладачами компіляторів, і з тими, хто конструює апаратне забезпечення, щоб з'ясувати, якими особливостями повинен володіти рівень команд. Якщо укладачі компіляторів вимагають наявності якоїсь особливості, яку інженери не можуть реалізувати, то така ідея не пройде.

Також, якщо розробники апаратного забезпечення хочуть ввести в комп'ютер будь-яку нову особливість, але укладачі програмного забезпечення не знають, як побудувати програму, щоб використовувати цю особливість, то такий проект ніколи не буде втілений. Після довгих обговорень і моделювання з'явиться рівень команд, оптимізований для потрібних мов програмування, який і буде реалізований.

Все це теорія. Факт сумісності програмного забезпечення змушує комп'ютерних розробників зберігати один і той же рівень команд в різних моделях або, принаймні, робити його зворотно сумісним. Зворотна сумісність це здатність нової машини виконувати старі програми без змін. Проте нова машина може містити нові команди і інші особливості, які можуть використовуватися новим програмним забезпеченням.

Добре розроблений рівень архітектури команд має величезні переваги перед поганим, особливо щодо обчислювальних можливостей і вартості. Продуктивність еквівалентних машин з різними рівнями команд може відрізнятися на 25%. Іноді з'являються нові рівні команд універсального призначення, а на спеціалізованих ринках (наприклад, на ринку

вбудованих систем або на ринку мультимедійних процесорів) вони виникають набагато частіше. Отже, важливо розуміти принципи розробки цього рівня.

Добра архітектура команд визначається на основі двох основних чинників:

По-перше, архітектура повинна визначати набір команд, які можна ефективно реалізувати в сучасній і майбутній техніці, що призводить до рентабельних розробок на кілька поколінь.

По-друге, архітектура команд повинна забезпечувати чітку мету для транслювання програми. Регулярність і повнота варіантів – важливі риси, які не завжди властиві архітектурі команд. Ці якості важливі для компілятора, якому важко зробити кращий вибір з кількох можливих, особливо коли деякі очевидні на перший погляд варіанти не дозволені архітектурою команд. Якщо говорити коротко, оскільки рівень команд є проміжною ланкою між апаратним та програмним забезпеченням, він повинен бути зручний і для розробників апаратного забезпечення, і для укладачів програмного забезпечення.

Щоб зробити програму рівня команд, укладач компілятора повинен знати, яка модель пам’яті використовується в машині, які регістри, типи даних і команди є в наявності і т. п. Вся ця інформація в сукупності і визначає рівень архітектури команд.

Для однієї архітектури рівень команд визначається формальним документом, який зазвичай випускається промисловим консорціумом, для іншої – ні.

Мета такого офіційного документа - дати можливість різним виробникам випускати машини даного конкретного виду, щоб ці машини могли виконувати одні і ті ж програми і отримувати при цьому одні і ті ж результати.

У таких документах містяться нормативні розділи, у яких викладаються вимоги, і інформативні розділи, які призначені для того, щоб допомогти читачеві, але не є частиною формального визначення. У нормативних розділах описані вимоги і заборони.

Інша важлива риса рівня архітектури команд полягає в тому, що в більшості машин є, принаймні, два режими. Привілейований режим призначений для запуску операційної системи. Він дозволяє виконувати всі команди. Призначений для користувача режим призначений для запуску програмних додатків. Цей режим не дозволяє виконувати деякі чутливі команди (наприклад, ті, які безпосередньо маніпулюють кеш-пам’яттю).

Для програмування на рівні архітектури команд використовується мова програмування асемблер. *Розроблення програм* мовою Асемблера передбачає два етапи.

1-й етап. Підготовка початкового тексту програми і оформлення його у вигляді текстового файлу (одного або декількох) за допомогою будь-якого редактора розширенням *.asm.*

2-й етап. Асемблювання і компілювання програми.

Структура програми має наступний вигляд:

```
include '%fasminc%/win32ax.inc'; підключення бібліотек Windows, які знаходяться в файлі  win32ax.inc або win64ax.inc, який підключається залежно від 32-х або 64-х розрядний процесор.
include 'win64ax.inc' ;цей файл знаходиться в папці D:\FASM\INCLUDE
.data           ;ініціалізовані дані
.data?          ;неініціалізовані дані
.const          ;константи
.code           ;виконуваний код
start:          ;початок виконуваного коду
.end start      ;мітка точки входу
```

Секція з даними, які ініціалізовані .data, включається до файлу, що виконується. Секція з неініціалізованими даними .data? не включається до файлу, що виконується та з'являється тільки тоді, коли програма завантажується до пам’ять. Секція .const містить константи. Секція .code містить виконуючий код програми. У кінці програми повинно стояти слово end, яке задає точку входу програми, тобто місце, з якого розпочинає виконуватися

програма. Секції .data, .data? мають повний доступ. Секції .const та .code мають атрибут доступу – тільки читання. Секція .const рідко зустрічається в програмах, оскільки константи можна задавати за допомогою макроозначень.

Дані в асемблері визначаються в секціях .data, .data? та .const. Місце під дані резервується за допомогою директив db, dw, dd, dq, dt.

```
db - 1 байт
dw - 2 байти - 1 слово
dd - 4 байти - 2 слова
dq - 8 байт - 4 слова
dt - 10 байт
```

Секція .data найуніверсальніша, резервує пам'ять під дані та відразу ж їх ініціалізує, тобто задає їм початкові значення. Усі дані з цієї секції включають до файла, який виконується. Секція .data? менш гнучка, оскільки дані не можна ініціалізувати. Усі дані цієї секції не включаються до файла, який виконується, тому місце тільки резервується, а початкові значення не задаються. Дані в обох секціях мають повний режим доступу. Секцію .const призначено тільки для читання. Але помилок не виникає під час спроби змінити ці дані (????).

```
     .data
Perem dd 0000FF11h
 X1 dw 01235h
Binary db 00111010b
 dd 11225599h
decimal dw 28d
large dq 01123456789ABCDEFh
 dw 1011100101010111b
 .data?
Perem1 dd ?
Perem2 dq ?
Perem5 dw ?
  ..const
Const1 dd 012345678h
  dw 768d
```

В асемблері можна задавати тільки ANSI рядки, Unicode рядки складніше задавати та для їх оброблення існує ціла низька API функцій.

Числа в асемблері може бути подано у двійковій, десятковій або шістнадцятковій системах. Щоб показати, у якій системі використано число, треба поставити після числа літеру. Для бінарної системи літеру b (приклад: 0000010b, 001011010b), для десяткової системи можна нічого не вказувати або літеру d (приклад: 4589, 2356d), для шістнадцяткової системи – літеру h та обов'язково писати на початку нуль (приклад: 00889h, 0AC45h, 056Fh).

Під час програмування під Win32 або Win64 є тільки WIN API, які експортують системні бібліотеки. API функція передає керування точці входу функції (точка входу функції це перша інструкція функції), сама функція знаходиться у пам'яті процесу всередині деякої бібліотеки. API функції знаходяться у бібліотеках kernel32.dll, user32.dll, gdi32.dll (класика) advapi32.dll і т.д. kernel64.dll, user64.dll, gdi64.dll, advapi64.dll відповідно, для того, щоб використовувати деяку функцію цих бібліотек треба спочатку завантажити потрібну бібліотеку до пам'яті, або включити необхідні функції до таблиці імпорту, тоді користувач завантажить бібліотеки. Бібліотека kernel32.dll є у кожному процесі, вона там існує до запуску та проєктується у пам'ять завжди за однаковими адресами. API-функції (application programming interface) помітно спрощують роботу в операційній системі. Додаткову інформацію про API - можна отримати або на MSDN або з будь-якої довідки Windows SDK, яка поставляється з будь-яким продуктом Borland.

Приклад 1:

```
include '%fasminc%/win32a.inc'; або include 'win64ax.inc'
entry start
section '.data' data readable writeable
Caption db 'Наша первая программа.',0
Text db 'Ассемблер на FASM — это просто!',0
section '.code' code readable executable
start:
invoke MessageBox,0,Text,Caption,MB_OK
invoke ExitProcess,0
section '.idata' import data readable writeable
library KERNEL32, 'KERNEL32.DLL',\
USER32, 'USER32.DLL'
import KERNEL32,\
ExitProcess, 'ExitProcess'
import USER32,\
MessageBox, 'MessageBoxA'
```

Перша команда програми це макроінструкція invoke. Вона викликає вбудовану в Windows API-функцію MessageBox. Після імені функції через кому йдуть її параметри. У функції MessageBox параметри такі:

– містить хендл вікна-власника. Хендл – це особистий номер, який видається операційною системою кожному об'єкту (процесу, вікну та ін.). 0 означає, що у вікна не має власника, воно не залежить ні від яких інших вікон;

– покажчик на адресу першої літери тексту повідомлення, що закінчується вищезазначеним нуль-термінатором. Щоб наочно зрозуміти, що це всього лише адреса, змістимо цю адресу на 2 байта прямо у виклику функції: invoke MessageBox, 0, Text + 2, Caption, MB_OK і переконаємося, що тепер текст буде виводитися без перших двох букв;

– покажчик адреси першої літери заголовка повідомлення;

– стиль повідомлення. Зі списком цих стилів ви можете ознайомитися, наприклад, в INCLUDE \ EQUATES \ USER32.INC.

Для реалізація в асемблері роботи процесора і оперативної пам'яті слугують команди які передають роботу АЛП, тобто роботу з регістрами. Команда `MOV` використовується для копіювання значення з одного місця до іншого. Це може бути регістр, елемент пам'яті або безпосереднє значення (тільки як початкове значення). Синтаксис команди:

`mov` приймач, джерело

Можна копіювати значення з одного регістра до іншого:

```
mov edx, ecx
```

Команда копіює зміст ecx до edx. Розмір джерела та приймача мають бути однаковими:

```
mov cl, byte ptr [34h]    ;cl набуде значення 0Dh
mov dx, word ptr [3Eh]    ;dx набуде значення 7DEFh
```

Префікс `ptr` означає, що треба брати з пам'яті деякий розмір. А префікс перед ptr позначає розмір даних:

- Byte – 1 байт,
- Word – 2 байти,
- Dword – 4 байти.

Іноді розмір можна не вказувати: `mov eax [00403045h].`

Оскільки eax – 32-розрядний регістр, Асемблер розуміє, що йому також потрібне 32-розрядне значення, у цьому разі з пам'яті зі зсувом 403045h.

Можна безпосередньо поміщати значення: `mov edx, 5006h`.

Ця команда запише до регістра edx, значення 5006. Дужки [і] використовують для набуття значення з пам’яті (у дужках знаходиться зсув), без дужок – це просто безпосереднє значення.

Можна використовувати регістр як елемент пам'яті (він має бути 32-розрядним у 32-розрядних програмах):

```
mov eax, 403045h          ;запише до eax значення 403045
mov cx [eax]              ;помістить до регістра CX значення (розміру word) ;з пам’яті, яка вказана в EAX (403045).
```

В останній команді процесор спочатку дивиться, яке значення (елемента пам’яті) містить eax, потім – яке значення знаходиться у цьому елементі пам’яті, а потім поміщає це значення (word, 16-бітний, тому що приймач cx, є 16-розрядним регістром) до `cx`.

Арифметичні операції – `ADD` (addition – додавання), `SUB` (substraction – віднімання), `MUL` (multiply – множення) та `DIV` (divide – ділення), мають такий синтаксис:

`add` приймач, джерело (приймач = приймач + джерело)

приймач	*джерело*	*приклад*
регістр	регістр	`add ecx, edx`
регістр	пам’ять	`add ecx, dword ptr [104h] / add ecx, [edx]`
регістр	значення	`add eax, 102`
пам’ять	значення	`add dword ptr [401231h], 80`
пам’ять	регістр	`add dword ptr [401231h], edx`

`sub` приймач, джерело (приймач = приймач – джерело)

приймач	*джерело*	*приклад*
регістр	регістр	`sub ax, bx`
регістр	пам’ять	`sub ecx, dword ptr [104h] / sub ecx, [edx]`

`mul` приймач (приймач = приймач × джерело)

Приймач	*Джерело*	*Приклад*
регістр	значення	`mov al, x0`
регістр	значення	`mul x1`
регістр	значення	`mov cx, 20h`
регістр	регістр	`mul cx`

Стекові операції – `PUSH, POP` використовуються для розміщення значення у стеку та вилучення його зі стека. Команда `PUSH` розміщує значення у стеку, тобто поміщає значення до комірки пам’яті, на яку вказує регістр ESP, після цього значення регістра ESP збільшується на 4. Команда `POP` витягує значення зі стека, тобто вилучає значення з комірки пам’яті, на яку вказує регістр ESP, після цього зменшує значення регістра ESP на 4. Значення, яке міститься у стеку останнім, вилучається першим. Під час переміщення значення до стека покажчик стека зменшується, а під час вилучення – збільшується.

Наприклад:

```
mov ecx, 100          ;помістити 100 до ecx
mov eax, 200          ;помістити 200 до eax
push ecx              ;збереження ecx, помістити значення з ecx (=100) до стека (розміщується першим)
push eax              ;помістити значення з eax (=200) до стека (розміщується останнім)
mov edx, ecx
pop ebx               ;вилучення значення зі стека до ebx: ebx набуде значення 200 (останнє розміщення, перше вилучення)
pop ecx               ;вилучення значення зі стека до ecx: ecx знову набуде значення 100 (перше розміщення, останнє вилучення)
```

Пам’ять, під час розміщення та вилучення значень у стеку:

Зсув	1203	1204	1205	1206	1207	1208	1209	120A	120B
Значення	00	00	00	00	00	00	00	00	00
					ESP				

(Стек тут заповнений нулями. ESP стоїть у тому місці, на яке він указує)

```
mov ax, 4560h
push ax
```

Зсув	1203	1204	1205	1206	1207	1208	1209	120A	120B
Значення	00	00	60	45	00	00	00	00	00
			ESP						

```
mov cx, FFFFh
push cx
```

Зсув	1203	1204	1205	1206	1207	1208	1209	120A	120B
Значення	FF	FF	60	45	00	00	00	00	00
	ESP								

```
pop edx
```

Зсув	1203	1204	1205	1206	1207	1208	1209	120A	120B
Значення	FF	FF	60	45	00	00	00	00	00
					ESP				

`edx` тепер `4560FFFFh`.

Символьні мітки. Одна з переваг асемблера полягає у використанні символьних міток і міток даних (змінних). Обидва типи міток – це символьні мітки. У секції з даними за допомогою директив db, dw, dd, dq резервуєте пам’ять під дані відповідного розміру. Для відкриття доступу до цих даних існують символьні мітки змінних. По суті, мітка повертає зсув, на якому вона знаходиться. Мітка віртуальна – у шістнадцяткових кодах у неї немає еквівалента, вона створена тільки для зручності програміста. Приклад:

```
.data
PARAM dd 11223344h
.code
mov eax, [PARAM]            ;непряма адресація
mov [PARAM], ebx
```

За визначенням це вірний метод використання міток. Після цих операцій регістр eax буде дорівнювати 11223344. У непрямій адресації адресою слугує вміст регістра, ім’я регістра беруть у квадратні дужки.

Розмір копійованих даних за замовчанням дорівнює розміру директиви, яка стоїть після мітки.

Символьні мітки відрізняються від міток даних тим, що їх можна описувати у будь-якому місці програми. Після символьної мітки треба ставити двокрапку. Вони також повертають зсув, на якому стоять. Вони використовуються з командою `call`. Команда `call` передає керування команді, яка знаходиться на зсуві, на який указує мітка.

Для отримання зсуву мітки є спеціальний префікс `offset`. Приклад використання префікса `offset`:

```
.data
PARAM1 dd 12345678h
.code
 mov eax, offset PARAM1
```

Після цього у регістрі eax буде знаходитися зсув мітки PARAM1. Під час використання префікса призначенням може бути елемент пам’яті, у наступних команд буде той же самий результат, що у першому випадку:

```
 mov [PARAM1], offset PARAM1
 mov eax [PARAM1]
```

Альтернативою префіксу `offset` є команда `lea`. Приклад:

```
  .data
 metka:
 PARAM1 dd 12345678h
  .code
 lea eax, metka
```

Після цього у регістрі eax буде знаходитися зсув мітки metka. У команді `lea` можна отримувати зсув мітки даних. Наступний приклад матиме той же самий результат, що і попередній:

```
  .data
 metka:
 PARAM1 dd 12345678h
  .code
 lea eax, PARAM1
```

У другому рядку значення x0 множиться на значення x1 та записується до регістра al, а в четвертому вміст регістра cx множиться сам на себе.

`div` дільник (eax = eax / дільник, edx = залишок)

Оскільки регістри можуть містити тільки цілочислові значення, результат ділення розбивається на частку та залишок. Залежно від розміру джерела, частка зберігається в eax, а залишок у edx:

Розмір джерела	Ділення	Частка у...	Залишок у...
BYTE (8-bits)	ax / дільник	AL	AH
WORD (16-bits)	dx:ax* / дільник	AX	DX
DWORD (32-bits)	edx:eax* / дільник	EAX	EDX

Наприклад: якщо dx = 2030h, а ax = 0040h, то dx/ax = 20300040h. dx/ax – значення dword, де dx є старшим word, а ax – молодшим. edx/eax – значення quadword (64 біти), де старше dword у edx, а молодше у eax.

Джерело операції ділення може бути:

1) 8-біт регістра (al, ah, cl...);
2) 16-біт регістра (ax, dx ...);
3) 32-біт регістра (eax, edx, ecx...);
4) 8-біт значення з пам’яті (byte ptr [xxxx]);
5) 16-біт значення з пам’яті (word ptr [xxxx]);
6) 32-біт значення пам’яті (dword ptr [xxxx]).

Джерело не може бути безпосереднім значенням, тому що тоді процесор не зможе визначити розмір початкового операнда.

Окрім операції додавання `ADD` є операція додавання `ADC`, яка додає значення перенесення до суми, використовується для арифметики підвищеної точності.

Логічні операції з бітами – `OR`, `XOR`, `AND`, `NOT`. Ці команди працюють з приймачем і джерелом, виняток команда 'NOT'. Кожний біт у приймачі порівнюється з тим же самим бітом у джерелі та, залежно від команди, 0 або 1 поміщається до біта приймача:

Команда	AND				OR				XOR				NOT	
Біт джерела	0	0	1	1	0	0	1	1	0	0	1	1	0	1
Біт приймача	0	1	0	1	0	1	0	1	0	1	0	1	X	X
Біт результату	0	0	0	1	0	1	1	1	0	1	1	0	1	0

`AND` (логічне І) установлює біт результату – 1, якщо обидва біти, біт джерела та біт приймача, установлено в 1.

`OR` (логічне АБО) установлює біт результату – 1, якщо один з бітів, біт джерела або біт приймача, установлено в 1.

`XOR` (НЕ АБО) установлює біт результату – 1, якщо біт джерела відрізняється від біта приймача.

NOT інвертує біт джерела.

Приклад 2:

```
mov ax, 3406d  ;ax = 3406 (десяткове), у двійковому – ;0000110101001110
mov dx, 13EAh  ;dx = 13EA (шістнадцяткове), у двійковому – 0001001111101010
xor ax, dx
```

Виконання операції XOR над цими бітами:

– приймач = 0000110101001110 (ax) ;
– джерело = 0001001111101010 (dx).

Результат = 0001111010100101 (нове значення у ax 7845 – у десятковому, 1EA5 – у шістнадцятковому).

Приклад 3:

```
mov ecx, FFFF0000h
not ecx
```

FFFF0000 – 11111111111111110000000000000000 у двійковому.

Якщо ви виконаєте інверсію кожного біта, то отримаєте: 00000000000000001111111111111111, у шістнадцятковому це 0000FFFF, означає після операції NOT, ecx міститиме 0000FFFFh.

Збільшення/Зменшення – INC/DEC. Ці команди збільшують або зменшують уміст пам'яті або регістра на одиницю. Приклад 4:

```
inc регістр                ;регістр = регістр + 1.
dec регістр                ;регістр = регістр – 1.
inc dword ptr [103405]     ;значення у [103405] буде збільшено на 1.
dec dword ptr [103405]     ;значення у [103405] буде зменшено на 1.
```

Команда порівняння – TEST. Команда TEST виконує операцію AND (логічне І) з двома операндами та залежно від результату встановлює або скидає відповідні прапори. Результат не зберігається. TEST використовують для перевірки бітів, наприклад у регістрі:

```
test eax, 100b
jnz  ;зсув
```

Команда JNZ виконує перехід, якщо у регістрі eax третій біт праворуч – установлений. Дуже часто команду test використовують для перевірки, чи дорівнює регістр нулю:

```
test ecx, ecx
jz   ;зсув
```

Команда JZ виконає перехід, якщо ecx = 0.

Команда – NOP. Ця команда нічого не робить (порожня команда). Вона тільки займає простір і час. Використовується для резервування місця у сегменті коду або організації програмної затримки.

Обмін значеннями – XCHG. Призначення: обмін двох значень між регістрами або між регістрами та пам'яттю:

```
mov eax, 237h
mov ecx, 978h
xchg eax, ecx
```

у результаті:

– eax = 978h;
– ecx = 237h.

Приклад 5, розглянемо умовні переходи на наданому коді:

```
1) mov eax,32  ;  в EAX 32h
2) mov ebx,33  ;  в EBX 33h
3) cmp eax,ebx  ;  порівнюємо EAX з EBX
4) jne metka  ;  якщо не рівні то стрибаємо на metka
```

```
   5) sub eax,eax  ; інакше обнуляємо EAX
metka:
   6) invoke ExitProcess,0 ; завершуємо процес
```

Якби в 4 пункті написали б не JNE, а JZ, то не стрибнули б на мітку, оскільки регістри EAX & EBX не рівні між собою.

ЗАВДАННЯ

1. Що знаходитиметься у регістрі EBX після того, як до нього помістять 32-розрядне значення зі зсувом: 1) 3F; 2) 35; 3) D?
2. Що знаходитиметься у регістрі AX, якщо до нього помістити 16-розрядне значення зі зсувом: 1) 34; 2) 39; 3) C?
3. Як зберігаються значення 50607080 і 1020 у пам'яті, чим вони вирізняються?
4. Змініть заголовок вікна і текст повідомлення в прикладі 1. Поясніть що ви зробили.
5. У приклад 5 додайте необхідні команди, щоб отримати повноцінну програму.

КОНТРОЛЬНІ ПИТАННЯ

1. Для чого призначена кеш-пам'ять?
2. Що таке тактова частота?
3. Що таке розрядність?
4. Для чого призначена оперативна пам'ять?
5. Для чого призначений контролер?
6. Для чого призначена загальна шина?
7. Скільки потрібно одиниць інформації для кодування одного символу?
8. Де записана програма BIOS комп'ютера?
9. Що відноситься до машинних системних інтерфейсів?
10. Які регістри належать до регістрів загального призначення і для чого вони використовуються?
11. Які регістри належать до регістрів сегменту та для чого вони використовуються?
12. Для чого потрібний регістр прапорів?
13. Для чого використовується стек?
14. У яких режимах можуть працювати процесори та чим вони вирізняються?
15. Що таке сегмент та офсет?
16. Які сегментні регістри ви знаєте, для чого вони використовуються?

РОЗДІЛ 3. РІВЕНЬ ОПЕРАЦІЙНОЇ СИСТЕМИ

3.1. Режими роботи ЕОМ

На рівні операційної системи (ОС) ЕОМ безпосередньо взаємодіє з користувацькими програмами, тобто виконує всі необхідні користувачу роботи пов'язані з обробкою інформації. Існує багатий спектр різноманітних ОС, який залежність від задач що виконує система (дивись розподіл ЕОМ за напрямками застосування), вони перш за все відрізняються різноманітними режимами роботи. Режим роботи це принцип структурної і функціональної організації апаратних і програмних засобів. У загальному випадку вони поділяють на однопрограмні і багатопрограмні.

В моделях першого і другого покоління використовувались *однопрограмні режими роботи*. У цих режимах основні ресурси ЕОМ (час роботи процесора, оперативна пам'ять і

ін.) цілком і повністю виділяються у використання користувача. *Однопрограмний режим поділяється на режим безпосереднього доступу і режим непрямого доступу*.

У *режимі безпосереднього доступу* користувач використовує ЕОМ для завантаження завдання і запускає його на виконання, спостерігає за ходом рішення і виведенням результатів. Після цього всі ресурси ЕОМ передаються у розпорядження іншого користувача (рис. 26(а)). Це неефективний режим роботи, який має низьку продуктивність завантаження технічних засобів, за рахунок того, що відбуваються простої процесора і це призводить до витрати часу на підготовку ЕОМ (включення, перевірка, завантаження ОС, введення завдань і т.д.) та через великий час реакції користувача.

Режим непрямого доступу це перший багатопрограмний режим при якому користувач не має безпосереднього доступу до ЕОМ. Він забезпечував більш повне завантаження процесора за рахунок принципу, який в подальшому ліг в основу багатопрограмної обробки. Під час виконання режиму підготовлені завдання користувачів складаються у пакет завдань. Процесор обслуговує програми користувачів у строгому порядку їх слідування в пакеті, виконання чергової програми не переривається до повного її завершення. Процесор виконує наступну програму тільки після завершення попередньої. Доступ користувача до ресурсів ЕОМ здійснюється засобами ОС, які і забезпечують автоматичний перехід від обслуговування одного завдання користувача до іншого та паралельну роботу пристроїв введення-виведення сумісно з процесором (рис. 26(б)), що дозволяє підвищити продуктивність ЕОМ. Цей режим ще називають послідовною пакетною обробкою.

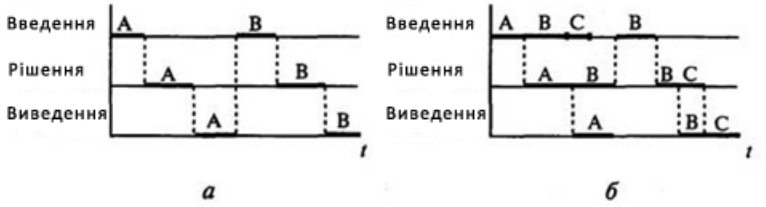

Рисунок 26. Однопрограмні режими роботи: а - режим безпосереднього доступу; б- режим непрямого доступу

Режим непрямого доступу не дозволяє повністю виключити випадки простою процесора або непродуктивного його використання. Випадок коли в програму заздалегідь не завантажені дані відображений на рис. 26(б), в цьому випадку спостерігається, на прикладі виконання завдання, як процесор змушене простоює. Неефективно працює ЕОМ і тоді, коли оброблювані програми захоплюють процесор на тривалий час, а інші програми пакета залишаються без обслуговування. Крім того у цьому режимі у ЕОМ відсутні можливості вирішення конфліктів, які пов'язані зі збоями програми під час роботи.

Багатопрограмний режим (мультипрограмний) роботи ЕОМ дозволяє одночасно обслуговувати кілька програм користувачів. Операційна система грає ключову роль забезпечення багатопрограмного режиму роботи. Основні завдання системи полягають в управлінні процесами, в розподілі ресурсів та забезпеченні безпеки. Основні етапи роботи ОС у цьому режимі:

1. Завантаження та розміщення програм у пам'яті. В цей етап входить завантаження кількох програм на оперативну пам'ять, таким чином щоб кожна програма розташовувалась у окремій області пам'яті. Під час роботи завантаженні програми отримують в своє розпорядження віртуальну область пам'яті, що запобігає впливу на інші програми і називається механізмом захисту пам'яті.

2. Управління процесами та планування. Під час цього етапу ОС для кожної програми створює процес і призначає їй визначений «час процесора». Процесом називається програма під час виконання. Існує кілька стратегій планування процесів:

– кругова черга (Round Robin) створюється так, що кожному процесу виділяється певний час виконання, якщо процес не завершено, він поміщається в кінець черги;

– пріоритетне планування полягає у тому, що процеси з вищим пріоритетом набувають більше процесорного часу;

– найкоротша робота, що залишилася, тоді пріоритет віддається процесам, які можуть швидше завершитися.

3. Перемикання контексту. В ході цього етапу ОС регулярно припиняє виконання одного процесу і перемикається на інший (це називається контекстне перемикання). При перемиканні контексту система зберігає поточний стан (реєстр процесора, покажчик інструкцій тощо) поточного процесу і завантажує стан наступного процесу, це забезпечує плавне виконання кількох програм «одночасно» (насправді вони по черзі отримують процесорний час).

4. Обробка операцій введення-виведення відбувається, коли програма виконує операцію введення-виведення (наприклад, читання файлу з диска), вона зупиняється, а ОС перемикає процесор на іншу програму. Під час цього етапу ОС управляє чергами операцій введення-виведення і повідомляє відповідний процес про завершення операції, щоб він міг продовжити виконання. Це продемонстровано на рис. 27.

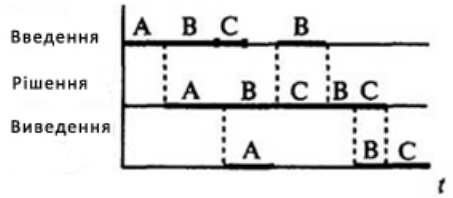

Рисунок 27. Багатопрограмний режим обробки

5. Управління пам’яттю операційна система забезпечує за допомогою технологій віртуальної пам’яті та сегментації. Якщо оперативної пам’яті недостатньо, ОС може тимчасово вивантажувати частину даних на жорсткий диск (сторінкова організація пам’яті) і повертати їх у оперативну пам’ять за необхідності.

6. Управління синхронізацією та виключенням взаємоблокувань (deadlock) відбувається при паралельній роботі кількох програм, коли два або більше процесів чекають на ресурси один одного. ОС може запобігати взаємним блокуванням за допомогою алгоритмів виявлення та запобігання deadlock (наприклад, алгоритм Дініца або механізм очікування по timeaut). Для синхронізації потоків використовуються семафори, м'ютекси та інші механізми.

7. Для забезпечення захисту та безпеки кожна програма виконується в «ізольованому середовищі», щоб вона не могла змінювати пам’ять інших програм. При цьому ОС контролює доступ до критично важливих ресурсів та захищає системні файли та налаштування.

Переваги багатопрограмного режиму полягають в наступному:

– ефективне використання процесора – поки один процес очікує на введення-виведення, інші продовжують виконуватися.

– збільшення продуктивності полягає в тому, що кілька програм можуть бути виконані паралельно, що прискорює загальний процес обробки даних.

– гнучкість та багатозадачність забезпечує користувача можливістю одночасно запускати та використовувати кілька програм (наприклад, текстовий редактор, браузер та програвач музики).

Недоліки багатопрограмного режиму полягають перш за все у складності управління, при цьому основне навантаження лягає на ОС, яка повинна керувати перемиканням контексту, синхронізацією процесів та унеможливлювати взаємні блокування. Під час роботи в такому режимі відбувається перевантаження пам’яті за рахунок надмірної кількості програм, що сповільнить роботу системи. Якщо процеси конкурують за ресурси (наприклад, процесор), це може спричинити затримки та призупинення у їх є виконанні.

Більш розвиненою формою багатопрограмної роботи ЕОМ є *режим поділу часу*. *Режим поділу часу (time-sharing)* – це метод мультипрограмування, при якому ресурси ЕОМ (особливо процесор) діляться між користувачами та завданнями таким чином, щоб кожному з них здавалося, що він працює із системою в індивідуальному режимі. До основних характеристик цього режиму відносяться:

– забезпечення «інтерактивної роботи» користувача із системою;

– встановлення кругового цикл обслуговування, під час якого процесор виділяє кожному процесу невеликі часові інтервали (кванти часу ъ), по завершенню яких процес припиняється, і керування передається іншому процесу (рис. 28).

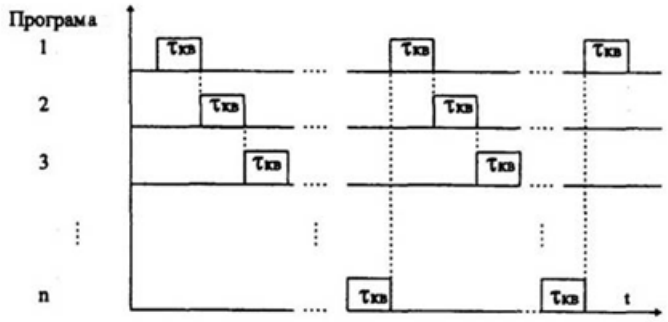

Рисунок 28. Режим поділу часу

– операційна система працює в режимі активного планування, тобто перемикає завдання не лише при очікуванні вводу-виводу, але й за таймером.

Все це створює у користувачів враження, що виконання їх програм відбувається одночасно. Наприклад робота сучасних персональних комп'ютерів, серверів та мобільних пристроїв, де користувачі можуть одночасно запускати браузери, редактори, програвачі відео тощо.

Вибір розміру часу кванта – критичне рішення, яке залежить від характеру завдань і системи. Занадто великий квант (ъ) зменшує частоту переключень контексту і система може виглядати «повільною» для користувача. При занадто малому кванті (ъ) більшість часу процесор буде зайнятий переключенням між процесами у результаті чого загальна продуктивність системи зменшується. Оптимальний квант (ъ) зазвичай вибирається так, щоб був компроміс між частотою переключення контексту та продуктивністю. У сучасних системах значення кванта може бути від 10 до 100 мс (мілісекунд).

Переваги режиму поділу часу полягають у швидкій чутливість системи і одночасній роботі кількох користувачів (розраховані на багато користувачів системи UNIX/Linux).

Недоліками є зниження продуктивності у результаті витрат на перемикання контексту (перемикання процесів) і складні механізми планування процесів

Складнішою формою поділу часу є *режим реального часу. Режим реального часу (real-time system, RTS)* – це режим роботи ЕОМ, при якому завдання повинні бути виконані в заздалегідь задані часові інтервали, інакше можливі серйозні наслідки.

Основною задачею цього режиму є забезпечення виконання критично важливих завдань у заданий час за рахунок надання пріоритетів усім завданням, які дозволяють завданням з високим пріоритетом переривати завдання з нижчим пріоритетом. Характеристикою режиму є жорсткі часові обмеження при яких час відгуку системи фіксований та передбачуваний. Обмеження повинні відповідати виду $t_P <= t_P^{доп}$ де t_P - час виконання завдання; $t_P^{доп}$ - допустимий час розв'язку.

На рис. 29 показана залежність вартості виконання завдання від часу t_P. При порушенні нерівності вартість рішення різко падає до нуля; в окремих системах вона може стати від'ємною, це показано штриховою лінією. Режим реального часу об'єднує практично всі системи, в яких ЕОМ використовується в контурі управління.

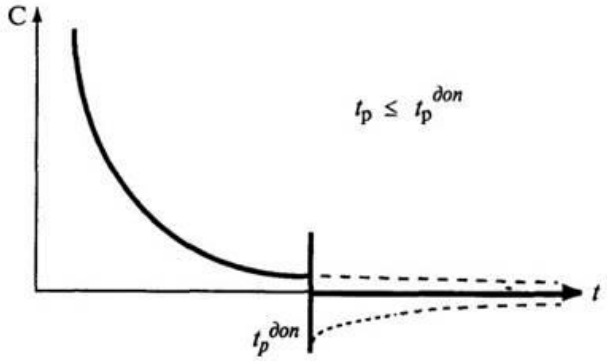

Рисунок 29. Залежність вартості рішення від часу в системах реального часу

Системи реального часу поділяються на два типи:

– жорсткі системи реального часу (hard real-time) з відмовою у дотриманні термінів виконання завдання, що може призвести до катастрофічних наслідків (наприклад у системах керування авіонікою літака);

– м'які системи реального часу (soft real-time) з порушенням тимчасових обмежень, що знижує якість роботи системи, але не призводить до критичних наслідків (наприклад, відтворення відео).

До системи реального часу відносяться системи керування промисловими роботами, автомобільні системи керування двигуном, авіоніка та системи керування повітряним рухом і т.д.

Переваги цього режиму полягають у швидкому відгуку та можливістю керувати «зовнішнім світом» у реальному часі а також у надійності та передбачуваності роботи системи.

До недоліків можна віднести необхідність у жорсткому контролі часу виконання завдань та високі витрати на розробку систем, оскільки необхідно забезпечити передбачуваність всіх процесів.

Похідні від останніх трьох режимів є режим багатозадачності, режим багато поточності, режим віртуалізації і режим розподілених обчислень.

Режим багатозадачності (multitasking) оснований на багатозадачності, як здатності системи одночасно обробляти кілька завдань. По суті, це покращений режим поділу часу з активним плануванням процесів та можливістю роботи користувача із системою в реальному часі. Багатозадачність поділяється на варіанти такі як кооперативна коли програми самі «поступаються» керуванню операційною системою (наприклад, старі системи Windows 3.x) і примусова коли ОС сама зупиняє виконання програм та перемикає завдання (як у сучасних Windows, Linux та macOS).

Режим багатопоточності (multithreading) оснований на багатопотоковості як здатності однієї програми запускати кілька «потоків» (частин програми), які можуть виконуватися паралельно, при цьому кожен потік може виконуватись незалежно від інших потоків і потоки одного процесу використовують загальну пам'ять та ресурси, що знижує витрати на їхнє перемикання порівняно з процесами. В якості приклада можна розглянути роботу у текстовому редакторі коли один потік обробляє текст, а інший – автозбереження файлу.

Режим віртуалізації (virtualization mode) оснований на віртуалізації яка дозволяє запускати кілька «віртуальних» комп'ютерів однією фізичної машині. Кожен віртуальний комп'ютер працює в режимі мультипрограмування або розподілу часу, але керується гіпервізором (спеціальним програмним забезпеченням для віртуалізації). Кожен віртуальний комп'ютер має власну операційну систему та ізольовані ресурси і підтримує паралельне виконання багатьох систем на одній апаратній платформі. В якості приклада можна навести роботу віртуальних машин у VMware, VirtualBox або хмарних платформ (AWS, Azure).

Режим розподілених обчислень працює у розподілених системах, які складаються у кілька обчислювальних вузлів (ЕОМ) і спільно обробляють завдання. Тут використовується гібридна модель поділу часу та мультипрограмування з можливістю передачі завдань між комп'ютерами. Кожен вузол системи може працювати незалежно та обробляти свої завдання, потім завдання можуть бути передані до іншого вузла, якщо він звільнився раніше. Прикладом можуть слугувати хмарні системи (Amazon AWS, Google Cloud) та розподілені обчислювальні проекти (BOINC).

3.2. Структура ОС Windows

Операційна система є базовою системною програмою. Зазвичай апаратно-програмне забезпечення типової обчислювальної системи представляють у вигляді набору шарів (рис. 30), при цьому операційній системі відповідає шар між обладнанням комп'ютера і іншим програмним забезпеченням. Таке розташування дозволяє ОС забезпечувати можливість раціонального використання устаткування комп'ютера зручним для користувача чином шляхом створення середовища для функціонування і розробки прикладних програм.

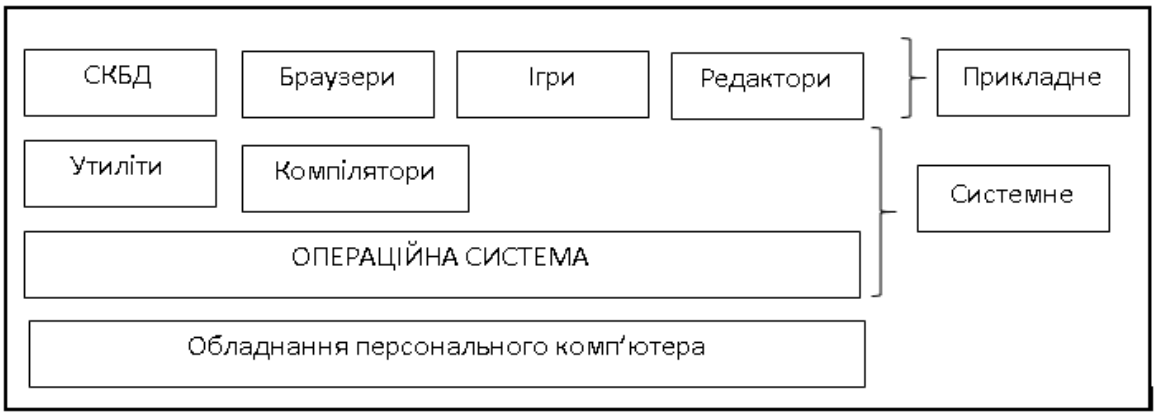

Рисунок 30. Шари програмного забезпечення комп'ютерної системи

Дружній інтерфейс між користувачем і комп'ютером досягається за рахунок абстрагування, яке є важливим методом спрощення, що дозволяє сконцентруватися на взаємодії високорівневих компонентів системи, ігноруючи деталі їх реалізації. У цьому сенсі про операційну система Windows кажуть, що це багатофункціональна, зручна у використанні та масштабована система, яка підтримує різноманітне програмне та апаратне забезпечення.

Іншими словами ОС можна уявити як менеджера ресурсів, що здійснює впорядкований і контрольований розподіл процесорів, пам'яті та інших ресурсів між різними програмами.

3.3. Історія створення ОС Windows

Операційні системи корпорації Microsoft можна умовно розділити на три групи:

- MS-DOS і MS-DOS + Windows 3.1,
- так звані споживчі (consumer) версії Windows (Windows 95/98 / Me)
- лінія ОС, які ведуть свій початок від Windows NT (Windows NT / 2000 / XP / Vista/...).

Однозадачна 16-розрядна ОС MS-DOS була випущена на початку 80-х років і потім широко застосовувалася на комп'ютерах з процесором x86. Спочатку MS-DOS була досить примітивна (деградація ОС), її оболонка займалася, головним чином, обробкою командного рядка, але в наступні версії було внесено багато покращень, запозичених, головним чином, з ОС Unix.

Потім під впливом успіхів дружнього графічного інтерфейсу корпорації Apple для комп'ютерів Macintosh була розроблена система Windows. Особливо широкого поширення набули версії Windows 3.0, 3.1 і 3.11.

Спочатку це була не самостійна ОС, а скоріше багатозадачна (з витісняючою багатозадачностюі) графічна оболонка MS-DOS, яка контролювала комп'ютер і файлову систему.

У 1995 р була випущена 32-розрядна ОС Windows 95, де була реалізована витісняюча багатозадачність. ОС Windows95 включала великий обсяг 16-розрядного коду, головним чином для забезпечення наступності з додатками MS-DOS. 16-розрядний код був присутній і в наступних версіях цієї серії Windows 98 і Windows Me. Іншою проблемою даної версії Windows, багато в чому обумовленої тією ж причиною, була нереєнтерабельність значної частини коду ядра. Так, якщо один з потоків був зайнятий модифікацією даних в ядрі, інший потік, щоб не отримати ці дані в суперечливому стані, змушений був чекати, тобто не міг скористатися системними сервісами. Це, найчастіше, зводило нанівець переваги багатозадачності.

ОС Windows NT (New Technology) - 32-розрядна ОС, сумісна з попередніми версіями Windows по інтерфейсу. Помітна спадкоємність в системі управління великим адресним простором і саморозміщувані безліччю процесу, в системі пріоритетів звичайних процесів і процесів реального часу, в засобах синхронізації і т.д. Разом з тим Windows NT – це абсолютно новий амбітний проект розробки системи з урахуванням новітніх досягнень в галузі архітектури мікроядра. Перша версія, названа Windows NT 3.1 для відповідності популярної Windows 3.1, була випущена в 1993р. Комерційного успіху добилася версія Windows NT 4.0, яка мала графічний інтерфейс Windows 95. На початку 1999р була випущена Windows NT 5.0, перейменована в Windows 2000. Наступна версія цієї ОС даної серії - Windows XP з'явилася в 2001 році, а Windows Server 2003 – в 2003р, далі випущена Windows Vista, раніше відома під кодовим ім'ям Longhorn. Windows 7 була випущена у 2009 році, включала напівпрозорий інтерфейсом Aero та нові функції, які включали підтримку мультитача, браузер Internet Explorer 8, покращену продуктивність і час запуску, Aero Snap (прив'язка вікна до краю екрана), Aero Shake (згортання вікна шляхом його тряски), підтримку віртуальних жорстких дисків, покращений Windows Media Center та покращені функції безпеки. Windows 8 була випущена у 2012 році, це повністю перероблена система з урахуванням використання сенсорного екрану для настільних комп'ютерів та ноутбуків з новим інтерфейсом плитки «Metro» та підтримкою x86-ПК і ARM-процессорів. Windows 10 випущена 2015 році включала такі нові функції, як Microsoft Edge, Cortana (Голосовий помічник), Windows Hello (розпізнавання відбитків пальців та обличчя) та ін. Всі подальші версії ОС мали накопичувальні оновлення для поліпшення інтерфейсу користувача, безпеки та інших функцій.

3.4. Особливості ОС Windows

Основні особливості ОС Windows є графічний інтерфейс, підтримка багатозадачності, безпека, сумісність із програмами та регулярні оновлення.

Графічний інтерфейс користувача (Graphical User Interface, GUI) одна з головних відмінних рис Windows. Він забезпечує зручність і простоту використання для користувачів, навіть якщо вони не мають досвіду роботи з комп'ютерами. Його особливостями є:

– робочий стіл, як основний простір для взаємодії користувача з файлами, ярликами та програмами;

– вікна, що забезпечують можливість кожному додатку працювати у своєму окремому вікні, яке можна змінювати за розміром, мінімізувати, закривати та переміщувати;

– меню «Пуск», що є центральною точкою доступу до програм, налаштувань і пошуку файлів;

– панель завдань, на якій відображаються відкриті програми, сповіщення та системний трей для швидкого доступу до функцій;

– контекстне меню, яке забезпечує швидкий доступ до дій над об’єктами за допомогою правої кнопки миші.

Підтримка багатозадачності Windows, що підтримує паралельне виконання кількох завдань**.** Це досягається завдяки розподілу часу і забезпечує продуктивність та ефективність використання системних ресурсів. В основні можливості багатозадачності входять:

– виконання декількох програм одночасно дозволяє працювати у текстовому редакторі, слухати музику та переглядати веб-сторінки одночасно;

– планувальник завдань забезпечує ефективний розподіл часу процесора між процесами;

– підтримка багато потоковості, що дозволяє програмам розділяти свої обчислення на кілька потоків, які виконуються паралельно.

Підтримка апаратного забезпечення Windows має широку підтримку драйверів пристроїв**,** що забезпечує простоту налаштування та підключення нового обладнання для роботи та надає наступні можливості:

– автоматичне виявлення пристроїв (Plug and Play), яке дозволяє Windows автоматично розпізнає нові пристрої та встановлює відповідні драйвери;

– підтримка периферійних пристроїв таких як миші, клавіатури, принтери, сканери, камери тощо;

– широка підтримка процесорів та чипсетів, це сумісність з архітектурами x86, x64 та ARM (в останніх версіях Windows 10/11).

Сумісність програмного забезпечення Windows підтримує велику кількість програмного забезпечення, це дозволяє запускати широкий спектр програм, що робить Windows універсальною платформою для роботи та розваг. Це сумісність з:

– Windows-додатками, що створюються для Windows (ігри, офісні програми, графічні редактори тощо);

– зворотна сумісність, це можливість запускати програми для старих версій Windows (наприклад, Windows 7) у сучасних версіях Windows (через «режим сумісності»);

– запуск додатків для Linux (WSL) у Windows 10/11 доступний Windows Subsystem for Linux (WSL) для роботи з Linux-додатками без віртуальної машини.

Система безпеки ОС Windows має вбудовані інструменти для захисту даних та безпеки системи, які захищають систему від вірусів, хакерських атак та несанкціонованого доступу. До основних механізмів захисту відносяться**:**

– Windows Defender – антивірусний захист у реальному часі, що запобігає проникненню шкідливих програм;

– брандмауер Windows фільтрує вхідний та вихідний трафік для захисту від зовнішніх атак;

– користувацькі облікові записи та права доступу дають можливість створення облікових записів із різними рівнями доступу (адміністратор, гість тощо);

– шифрування BitLocker дозволяє шифрувати диски для захисту даних від несанкціонованого доступу.

Система оновлень (Windows Update) це автоматична система для виправлення помилок, оновлення драйверів і підвищення рівня безпеки, що забезпечує підтримку актуальної версії ОС та забезпечення безпеки системи. В систему оновлень входять:

– автоматичне завантаження оновлень, що позбавляє користувач у необхідності вручну шукати оновлення;

– можливість відкладання оновлень, ця функція є тільки у корпоративних версіях для зменшення ризику збоїв у роботі;

– регулярні оновлення безпеки дозволяють усувати вразливості, що з’являються в системі.

Підтримка віртуалізації, ця технологія дозволяє запускати інші ОС або програми в ізольованих середовищах, що підвищує безпеку при роботі з підозрілими файлами та програмами. В неї входять:

– Hyper-V технологія віртуалізації для запуску віртуальних машин на Windows (доступна у версіях Pro та Enterprise);

– пісочниця Windows (Windows Sandbox) – ізольоване середовище для безпечного запуску підозрілих програм.

Інструменти адміністрування та налаштування Windows, які дозволяють налаштовувати ОС під потреби користувача та адміністратора. До них відносяться:

– диспетчер завдань, що дозволяє переглядати запущені програми, процеси та їхній вплив на ресурси системи;

– редактор групової політики доступний у версіях Windows Pro та Enterprise для керування налаштуваннями користувачів;

– редактор реєстру (Regedit) дозволяє змінювати внутрішні параметри системи;

– панель управління та налаштування до якої входять інструменти для налаштування параметрів системи, таких як час, мова, мережа тощо.

3.5. Архітектура ОС Windows

Архітектура ОС Windows побудована за багаторівневим принципом з чітким поділом між режимом користувача та режимом ядра. Кожен рівень (шар) відповідає за виконання певних функцій, що дозволяє зробити систему більш модульною, універсальною, безпечною та зручною для оновлення та обслуговування. Взаємодія між рівнями забезпечується за допомогою драйверів, підсистем та інтерфейсів API.

Архітектура ОС, зазнала ряд змін в процесі існування і вдосконалення. Перші версії системи мали мікроядерний дизайн, архітектура пізніших версій системи вже не є мікроядерною. В результаті в ядрі ОС Windows переплетені елементи мікроядерної архітектури та елементи монолітного ядра (гібридне ядро).

Архітектура Windows складається з трьох основних рівнів:

1. Рівень користувача (User Mode)
2. Рівень ядра (Kernel Mode)
3. Апаратний рівень (Hardware Layer)

Кожен із цих рівнів має свої підсистеми та компоненти, які забезпечують роботу ОС та взаємодію між користувачем, програмами та обладнанням.

Спрощена схема архітектури, орієнтована на виконання Win32/64-додатків, показана на рис. 31.

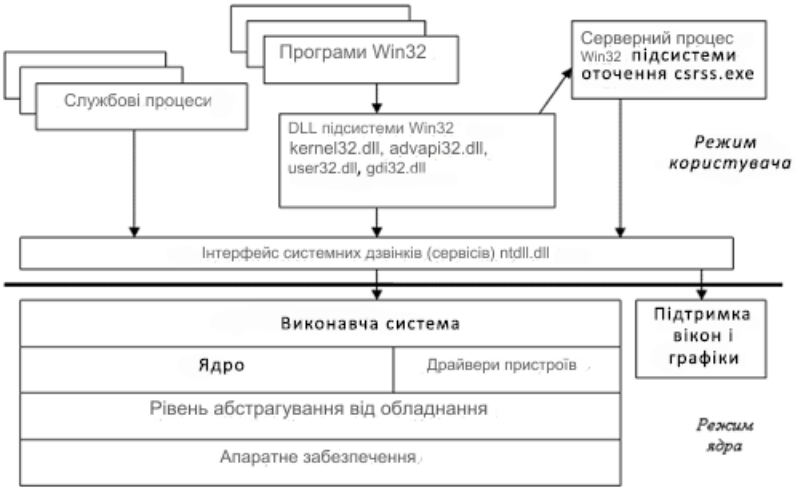

Рисунок 31. Спрощена архітектурна схема ОС Windows

ОС Windows складається з компонентів, які працюють в режимі ядра, і компонентів, які працюють в режимі користувача. На схемі чітко проглядаються кілька функціональних рівнів, кожен з яких користується сервісами більш низького рівня.

Апаратний рівень (Hardware Layer) це найнижчий рівень, що включає фізичне обладнання комп'ютера (процесор, пам'ять, пристрої вводу-виводу, мережеві інтерфейси тощо). Він забезпечує фізичну базу для виконання програм, але самостійно не може виконувати програми без програмного забезпечення.

Рівень абстрагування від обладнання або рівень абстракції апаратного забезпечення (Hardware Abstraction Layer, HAL) це програмний інтерфейс, що створює проміжний шар між апаратним забезпеченням та ядром операційної системи що забезпечує переносимість ОС і зменшує залежність системи від конкретного обладнання. Цей рівень забезпечує:

– абстракцію пристроїв для того, щоб ОС Windows могла працювати з різними типами обладнання (різні процесори, материнські плати тощо), завдяки тому, що HAL приховує специфічні деталі роботи з обладнанням;

– уніфікований доступ до пристроїв через HAL вони взаємодіють із драйверами та ядро системи;

– переносимість ОС відбувається завдяки HAL ОС Windows можна адаптувати до роботи на різних платформах.

Ядро операційної системи (Kernel) це центральна частина ОС, яка забезпечує взаємодію між апаратним та програмним забезпеченням та забезпечує основні функції для управління ресурсами системи, керування процесами, пам'яттю, введенням-виведенням і безпекою. До основних компонентів ядра відносяться:

– планувальник (Scheduler) керує розподілом процесорного часу між процесами та потоками;

– керування пам'яттю забезпечує віртуальну пам'ять і керування фізичною пам'яттю (RAM);

– керування файлами (File System Manager) забезпечує доступ до файлових систем (NTFS, FAT32 тощо);

– керування пристроями (I/O Manager) організовує введення-виведення між процесами та пристроями через драйвери;

– механізм синхронізації підтримує багатозадачність та забезпечує синхронізацію потоків.

Режим ядра (Kernel Mode) це режим роботи процесора, де він має повний доступ до всієї пам'яті та пристроїв і обробляє критично важливі системні виклики. У цьому режимі працюють ядро операційної системи та всі драйвери. Це відбувається завдяки системним драйверам (Kernel Drivers), що забезпечують доступ до пристроїв (відеокарти, диски тощо), механізмам обробки переривань, які дозволяють обробляти події від пристроїв і службам системи – внутрішнім сервісам для виконання системних функцій.

Драйвери пристроїв (Device Drivers) це спеціальні програми, які дозволяють операційній системі взаємодіяти з апаратними пристроями (принтерами, відеокартами, мережевими картами тощо) через уніфікований інтерфейс. Вони поділяються на драйвери режиму ядра (Kernel-Mode Drivers), що мають повний доступ до апаратних ресурсів та драйвери режиму користувача (User-Mode Drivers), що працюють у середовищі користувача та забезпечують більшу стабільність (наприклад, драйвери принтерів).

Режим користувача (User Mode) це частина операційної системи, де працюють усі програми користувача. Процеси в режимі користувача мають обмежений доступ до пам'яті та пристроїв, і вони можуть виконувати лише обмежену кількість системних викликів, які забезпечують середовище для виконання програм користувача та взаємодії з ОС через системні виклики та API. Основними компонентами цього режиму є**:**

– програми користувачів, які запускає користувач (браузери, редактори текстів тощо);

– динамічні бібліотеки DLL (Dynamic Link Libraries), які спільно використовуються різними програмами (наприклад, kernel32.dll, user32.dll);

– API WinAPI – інтерфейс прикладного програмування, який дозволяє програмам звертатися до системних функцій (відкриття файлів, створення процесів тощо).

Підсистеми операційної системи Windows (Subsystems), що дозволяють запускати програми, написані для різних середовищ (Unix, 32-bit, 64-bit, Linux тощо). До них відносяться:

– Win32 (основна підсистема) дозволяє запускати програми Windows (використовується для більшості сучасних програм);

– POSIX-підсистема забезпечує підтримку програм UNIX/Linux (раніше використовувалася, але зараз замінена WSL);

– Windows Subsystem for Linux (WSL) – сучасний механізм для запуску програм Linux у середовищі Windows;

– WOW64 (Windows on Windows 64-bit) — дозволяє запускати 32-бітні програми на 64-бітній системі.

Основні компоненти ОС Windows реалізовані в системних файлах, що знаходяться в каталозі system.

3.6. Підсистема Win32 і Win64

Взаємодія між додатком і операційною системою здійснюється за допомогою системних викликів (системних сервісів в термінології Microsoft). Однак програма не може викликати системний виклик безпосередньо (більш того, системні виклики не задокументовані). Замість цього додаток повинен скористатися програмним інтерфейсом ОС - Win32 API.

Win32 API (Application Programming Interface) – основний інтерфейс програмування в сімействі операційних систем Microsoft Windows. Функції Win32 API, наприклад, CreateProcess() або CreateFile(), – документовані, що викликаються підпрограми, реалізовані підсистемою Win32.

До складу підсистеми Win32 входять:

– серверний процес підсистеми оточення csrss.exe,

– драйвер режиму ядра Win32k.sys,

– dll - модулі підсистем (kernel32.dll, advapi32.dll, user32.dll і gdi32.dll), які експортують Win32-функції і драйвери графічних пристроїв.

В процесі еволюції структура підсистеми зазнала змін. Наприклад, функції вікон і малювання з метою підвищення продуктивності були перенесені з серверного процесу, що працює в режимі користувача, в драйвер режиму ядра Win32k.sys. У Win32® API використовується *плоска* 32-розрядна модель пам’яті. Кожному процесу виділяється «особисто» (private) ізольований адресний простір, розмір якого складає 4Gb. Цей простір розбивається на регіони. У загальному можна сказати, що нижні 2Gb це простір відведений процесу для вільного використання, а верхні 2Gb зарезервовані для використання операційною системою.

Win64-код об’єднує в собі основні можливості 32х-розрядного коду, а також включає зміни, пов’язані з підвищенням розрядності. Він налічує:

- 64х-розрядні покажчики;
- 64х-розрядні типи даних;
- 32х-розрядні типи даних;
- інтерфейс Win64 API.

В Win64 32х-розрядні типи даних не зникли при підвищенні розрядності платформи, а 32х-розрядні покажчики зникли повністю тому що основною для програмування 32х-розрядних додатків використовується 32х-розрядні покажчики, що дає можливість створення сегментів пам’яті об’ємом до 4 Гбайт. 64х-розрядні покажчики забезпечують можливість

адресації до 16 Тбайт пам’яті (1 Тбайт = 1012 Мбайт). Сучасними бізнес-додатками цей обсяг цілком використовується.

Функції в Win64 API зазнали незначних змін. Тільки назви деяких з них були змінені так, щоб відобразити приналежність до 64х-розрядної платформи. У більшості випадків змінам піддалися лише типи параметрів, які є аргументами виклику функцій. Всі інші переваги (можливість відмовитися від використання файлів підкачки і т. д.) пов’язані або із збільшенням об’єму адресації, або з новими типами даних.

Всі налаштування і конфігурація операційної системи знаходяться в реєстрі Windows, який дозволяє при вірному налаштуванні підвищити продуктивності системи. Реєстрі також зберігає контейнер, який містить ключі та спеціальні параметри розділів реєстру. Windows моє вбудований редактор реєстру, що використовується для перегляду та зміни параметрів. Перед внесенням змін до реєстру необхідно зробити копію реєстру, яка дає гарантію при виниканні проблем після змін, їх можна буде скасувати.

Доступ до редактора реєстру можна здійснити натисненням клавіш ***Win + R***, ввести "regedit" і підтвердити цю дію. Для того щоб зробити копію необхідно перейти до Файл -> Експорт, встановивши Діапазон експорту на ***All.*** Зберігати резервну копію бажано в безпечному місці.

З іформацією про розділи реєстру можна ознайомитись на сайті https://learn.microsoft.com.

Якщо необхідно прискорити роботу систему можна вимкнути анімацію Windows або запровадити налаштування затримки показу меню. Для цього перейдіть до «HKEY_CURRENT_USER\Control Panel\Desktop». Щоб вимкнути анімацію двічі клацніть 'WindowMetrics' і встановіть значення 'MinAnimate' на **0**, ця дія вимикає анімацію та може покращити реакцію системи. Щоб здійснити затримку показу меню необхідно змінити значення 'MenuShowDelay', нижче значення призведе до помітного прискорення.

ЗАВДАННЯ

1. Створіть резервну копію реєстру за допомогою експорту у файл.
2. Ознайомтесь з інформацією про розділи реєстру і напишіть які в них функції .
3. Оптимізуйте роботу системи за допомогою зміни параметрів відображення анімації і показу меню.
4. Відновіть реєстр зі зробленої попередньо копії.

КОНТРОЛЬНІ ПИТАННЯ

1. Поясніть суть режиму непрямого доступу?
2. В чому полягає особливість багатопрограмного режиму роботи ЕОМ?
3. Яку роль відіграє пакет в режимі класичного мультипрограмування, як він працює?
4. Як відбувається обробка в режим поділу часу, Яке призначення кванту часу?
5. Назвіть складові архітектури ОС Windows, для чого вони призначені?
6. В чому різниці між підсистемами Win32 і Win64?
7. Для чого призначений реєстрі Windows?

РОЗДІЛ 4. МЕРЕЖЕВИЙ РІВЕНЬ

Мережа - це сукупність ресурсів, призначених для передачі та обробки даних. В якості ресурсів можуть виступати:

- файлові ресурси – спільні папки та файли на сервері або комп'ютерах мережі;

– принтери та сканери – пристрої, доступні для спільного використання користувачами;

– програми та програмне забезпечення – доступ до віддалених додатків (через термінальні сервери або віртуальні машини);

– бази даних – спільний доступ до баз даних через клієнт-серверні архітектури;

– мережеві служби – доступ до серверів електронної пошти, веб-серверів, FTP-серверів тощо;

– обладнання – доступ до маршрутизаторів, комутаторів та інших мережевих пристроїв для налаштування або адміністрування.

Міжнародна організація по стандартизації визначила обчислювальну мережу *як послідовну біт-орієнтовану передачу інформації між пов'язаними один з одним незалежними пристроями.*

Об'єднані в мережу комп'ютери обмінюються інформацією, спільно використовують периферійне устаткування включаючи: принтери, плотери, дискові накопичувачі, сканери та інше, а також інформаційні ресурси: каталоги, файли, прикладні програми, ігри, бази даних та інше.

Комп'ютер користувача або програмне забезпечення (додаток), що надсилає запити на сервер, обробляє та відображає отримані дані користувачу називатиметься – *клієнт (client).* *Сервером* називається пристрій або програмне забезпечення, що приймає запити від клієнтів, обробляє їх та надсилає відповіді, його завдання зберігання, обробка та управління даними, обслуговування клієнтських запитів. *Канали зв'язку* (data link) створюються по лініях зв'язку за допомогою мережевого устаткування і фізичних засобів зв'язку. Між взаємодіючими інформаційними системами через фізичні канали комунікаційної мережі і вузли комутації встановлюються *логічні канали. Логічний канал* – це шлях для передачі даних від однієї системи до іншої. Логічний канал прокладається по маршруту в одному або декількох фізичних каналах. Логічний канал можна охарактеризувати, як маршрут, прокладений через фізичні канали і вузли комутації.

Інформація в мережі передається *блоками даних* по процедурах обміну між об'єктами. Ці процедури називають *протоколами передачі даних. Протокол* – це сукупність правил, що встановлюють формат і процедури обміну інформацією між двома або декількома пристроями.

Завантаження мережі характеризується параметром, що називається *трафіком. Трафік* (traffic) – це потік повідомлень в мережі передачі даних. Під ним розуміють кількісний вимір у вибраних точках мережі переданого числа блоків даних і їх довжини, що вимірюється у бітах в секунду (бот).

Метод доступу – це спосіб визначення того, яка з робочих станцій зможе наступною використати канал зв'язку і як управляти доступом до каналу зв'язку (кабелю).

4.1. Класифікація комп'ютерних мереж

Комп'ютерні мережі можна класифікувати за різними критеріями: за розміром, топологією, типом переданих даних, технологією доступу та іншими параметрами. Нижче розглянуті основні класифікації комп'ютерних мереж.

Класифікація за розміром і масштабом здійснюється за географічним розташуванням та розміром комп'ютерних мереж. Це визначає, які відстані вони можуть покривати та як багато пристроїв можуть бути підключені до мережі. За цією ознакою мережі поділяються на:

– *LAN (Local Area Network) локальні мережі* в них об'єднані комп'ютери та пристрої, які знаходяться в межах обмеженого простору, наприклад, в одному будинку, офісі або на підприємстві. Розмір в такій мережі покриває обмежену територію (до кількох км). Вони мають високу швидкість передачі даних (від 100 Мбіт/с до 10 Гбіт/с).

– *MAN (Metropolitan Area Network) міська мережа*, яка покриває більшу територію, ніж LAN, але меншу за розміри, ніж WAN, зазвичай це мережа на рівні міста, використовується для об'єднання кампусів, міські телекомунікаційні мережі, її розмір до 50 км, часто має швидкість до 10 Гбіт/с.

– *WAN (Wide Area Network) глобальна мережа*, що охоплює великі географічні області, може поєднувати різні міста, країни та континенти. Вона може використовувати для з'єднання комунікаційні канали зв'язку через супутники, оптоволокно тощо. Розмір цих мереж охоплює великі відстані (від кількох десятків км до глобальних мереж). Швидкість може бути дуже змінною залежно від технологій і стандартів (від кількох кбіт/с до кількох Гбіт/с). Прикладом таких мереж можуть бути Інтернет, корпоративні мережі, з'єднання між філіями компаній.

– *PAN (Personal Area Network) персональна мережа* це дуже маленька мережа, яка об'єднує пристрої в межах однієї людини, зазвичай в межах кількох метрів. Розмір такої мережі зазвичай до 10 метрів. Швидкість зазвичай від 1 Мбіт/с до 3 Мбіт/с. Це Bluetooth-з'єднання між мобільним телефоном та гарнітурою, бездротові миші та клавіатури.

– *GAN (Global Area Network) глобальна мережа*, яка об'єднує глобальні ресурси та дозволяє здійснювати передачу даних через континенти. Розмір її охоплює континенти, забезпечуючи глобальне з'єднання. Це Інтернет та підключення супутників.

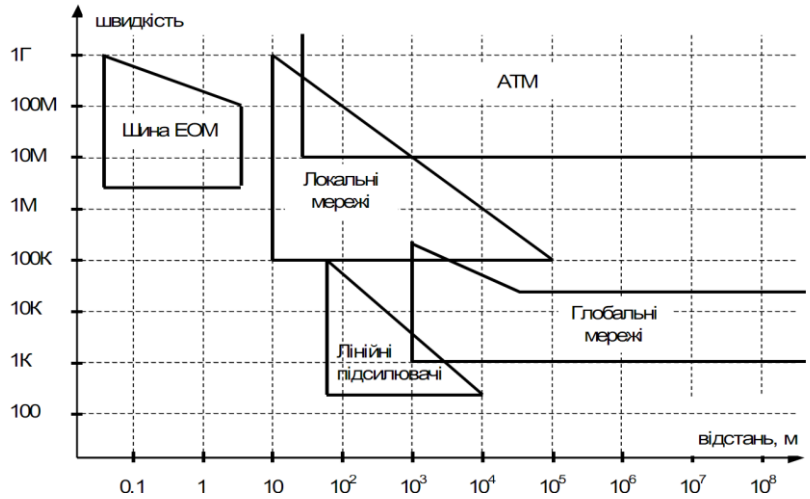

Рисунок 32. Діаграмам "відстань - швидкість"

Співвідношення між різними типами мереж найкраще простежити на діаграмі "відстань - швидкість" (рис.32).

Мережі ATM (Asynchronous Transfer Mode режим асинхронного передавання) відносяться до класу мереж MAN.

Класифікація за топологією (структурою мережі) визначається відповідно певній фізичній структурі з'єднання усіх комп'ютерів. *Топологія мережі* – це спосіб організації та з'єднання вузлів комп'ютерної мережі (пристроїв, таких як комп'ютери, сервери, маршрутизатори, комутатори тощо) за допомогою фізичних або логічних зв'язків. Топологія визначає, як передаються дані між пристроями, які маршрути вони використовують та як взаємодіють між собою. Вибір топології залежить від кількості пристроїв, чим більше пристроїв використовується, тим більш складна топологія потрібна; продуктивності мережі, для великих навантажень зазвичай обираються більш надійні топології (наприклад, зіркоподібна або повнозв'язна); витрат на створення мережі так прості топології, як-от шина або кільце, дешевші, але менш надійні. До топологій мереж належать наступні види:

1. Шинна топологія (Bus), в якій усі пристрої підключені до єдиного загального каналу передачі даних (шини). Її переваги полягають в простоті підключення нових пристроїв і низька вартість побудови, але при цьому вона уразлива до збоїв шини і має

низьку продуктивність при високому навантаженні. На цій топології будувались старі Ethernet-мережі.

2. Зіркоподібна топологія (Star) , в якій усі пристрої підключені до центрального вузла (наприклад, комутатора або концентратора). В ній легко додавати або видаляти пристрої, а також просте виявлення несправностей, але вона залежність від центрального вузла – якщо він виходить з ладу, мережа перестає працювати. На цій топології побудована більшість сучасних локальних мереж (LAN).

3. Кільцева топологія (Ring), в якій пристрої з'єднані один з одним у вигляді кільця, де кожен вузол з'єднаний з двома сусідніми вузлами. Завдяки передачі даних у чітко визначеному порядку, в ній відсутні колізій (за певних умов), але поломка одного вузла або з'єднання може зупинити роботу всієї мережі. Використовується у мережах Token Ring.

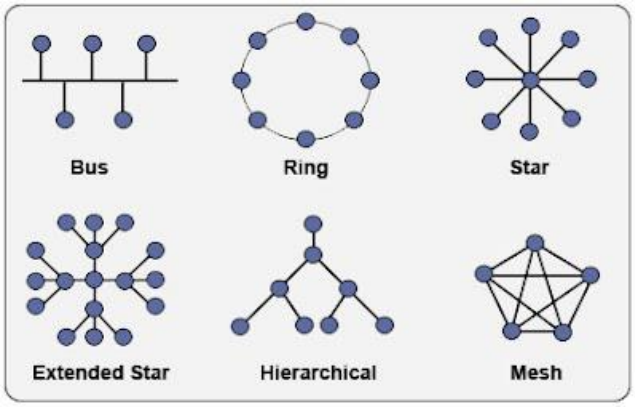

Рисунок 33. Топології комп'ютерних мереж.

4. Деревоподібна топологія (Tree) має ієрархічну структуру, що поєднує кілька зіркоподібних топологій через головний вузол. Вона має легку масштабованість і просту структуровану організація, але проблеми у головному вузлі можуть вплинути на всю мережу. Використовується в ієрархічних корпоративних мережах.

5. Повнозв'язна топологія (Mesh), в якій кожен пристрій з'єднаний з усіма іншими пристроями мережі. Ця мережа має висока відмовостійкість – навіть якщо один вузол або лінія вийде з ладу, дані все одно можуть бути передані іншим маршрутом, але складна і має високу вартість з'єднань. Використовується в критично важливих мережах (військові, фінансові мережі, мережі центрів обробки даних).

6. Гібридна топологія поєднує кілька різних топологій у межах однієї мережі. Має гнучкість у побудові мережі та можливість створення оптимальної архітектури, але складне управління та налагодження. Використовується в великих корпоративних мережах, де одночасно використовуються зіркоподібна та деревоподібна топології.

Класифікація за типом переданих даних визначає два типи мереж, які передають різні типи даних: текст, аудіо, відео тощо і можуть бути використані для конкретних потреб, таких як обмін великими файлами або відео конференції та мережі для передачі голосу (VoIP) це спеціалізовані мережі, які дозволяють передавати голосові повідомлення через Інтернет за допомогою технології Voice over IP (VoIP).

Класифікація за технологією доступу, ці технології призначені для підключення до мережі і поділяються на дротові (Ethernet, оптоволокно), які використовують фізичні кабелі (мідні, оптичні) для передачі даних між пристроями та бездротові (Wi-Fi, 4G, 5G, супутниковий інтернет), які дозволяють передавати дані без використання фізичних кабелів. Вибір технології залежить від швидкості, відстані та потреб користувача.

До дротових технологій відносяться:

1. Ethernet (LAN - локальна мережа) найпоширеніша технологія для підключення пристроїв у локальних мережах (LAN) в офісах, будинках, школах, на підприємствах, використовує виту пара (UTP, STP) та оптичне волокно, має швидкість передачі даних 10 Мбіт/с, 100 Мбіт/с (Fast Ethernet), 1 Гбіт/с (Gigabit Ethernet), 10 Гбіт/с і більше (10G Ethernet). В основі технології лежить метод доступу CSMA/CD (прослуховування каналу і виявлення колізій), вона використовує MAC-адреси для ідентифікації пристроїв.

2. Оптоволоконний доступ (FTTx) – технологія передачі даних за допомогою оптичного волокна, що забезпечує високошвидкісний доступ, надійність, можливість передачі великих обсягів даних і використовується Інтернет-провайдерами для підключення

будинків, офісів і підприємств. Має типи підключення FTTH (Fiber To The Home) – оптичне волокно до будинку користувача, FTTB (Fiber To The Building) – оптичне волокно до будівлі, а далі — вита пара до користувачів та FTTC (Fiber To The Curb) — оптичне волокно до "бордюру" або вуличного комутатора, а далі — вита пара до будинку, швидкість передачі в

3. DSL (Digital Subscriber Line) використовує телефонні ліній (мідні проводи) для високошвидкісної передачі даних, доступ відбувається через існуючі телефонні лінії, але швидкість залежить від відстані до АТС (автоматичної телефонної станції). Ця технологія використовується у домашніх мережах для підключення до інтернету через телефонні лінії та має швидкість від 128 Кбіт/с до 100 Мбіт/с (залежно від відстані) і тип підключення ADSL (Asymmetric DSL) – асиметричний доступ (швидкість завантаження вища, ніж швидкість вивантаження) й VDSL (Very high-speed DSL) – більш швидкісна версія DSL (до 100 Мбіт/с).

4. Кабельний інтернет (Coaxial Cable) використовує коаксіальний кабель, що традиційно використовується для кабельного телебачення та для передачі даних при цьому має високу швидкість завантаження, але продуктивність може знижуватися, якщо багато користувачів підключено одночасно. Цю технологію використовують Інтернет-провайдери, які надають послуги інтернету та кабельного телебачення, вона має швидкість від 10 Мбіт/с до 1 Гбіт/с і базується на DOCSIS (Data Over Cable Service Interface Specification).

5. Powerline (PLC - Power Line Communication) використовує електричнй проводів для передачі інтернет-сигналу, при чому передача інтернету відбувається через стандартну електричну розетку зі швидкістю від 200 Мбіт/с до 1 Гбіт/с. Використовується ця технологія у будинках та офісах для розширення зони Wi-Fi або підключення пристроїв у важкодоступних місцях.

До бездротових технологій відносяться:

1. Wi-Fi (Wireless Fidelity) локальна бездротова мережа для підключення пристроїв до інтернету зі зручним підключенням без кабелів, але швидкість залежить від відстані до роутера, працює зі швидкістю від 54 Мбіт/с до 9,6 Гбіт/с (Wi-Fi 6) за стандартами 802.11a/b/g/n/ac/ax (Wi-Fi 6), використовується дома, у офісі, і громадських місцях.

2. Мобільні мережі (3G, 4G, 5G) для доступу до інтернету у будь-якому місці зі швидкістю для 3**G** — до 42 Мбіт/с, для 4G (LTE) — до 300 Мбіт/с і для **5G** — до 10 Гбіт/с, може використовуватися в мобільних пристроях (телефони, планшети) та точки доступу Wi-Fi (роутери)

3. WiMAX (Worldwide Interoperability for Microwave Access) – бездротовий широкосмуговий доступ до інтернету у віддалених місцях, використовується для підключення віддалених населених зі швидкістю до 70 Мбіт/с.

4. Bluetooth – бездротова технологія для передачі даних між пристроями на короткій відстані (до 10-100 метрів), використовується для з'єднання пристроїв (навушники, клавіатури, миші) зі швидкістю передачи даних до 24 Мбіт/с (Bluetooth 5.0).

5. Супутниковий інтернет забезпечує доступ до інтернету через супутниковий зв'язок, працює у віддалених місцях, на морських суднах, в авіації, забезпечує доступ у будь-якій точці світу зі швидкістю від 10 Мбіт/с до 150 Мбіт/с (Starlink до 1 Гбіт/с).

Класифікація за типом доступу до ресурсів основана на моделях доступу, існує кілька популярних моделей організації доступу до ресурсів у мережі. Кожна з них має свої унікальні особливості та підходи до обміну даними та доступу до ресурсів. Кожна модель підходить для конкретних завдань, наприклад P2P для файлообміну, хмарна — для роботи з великими обчислювальними завданнями, віддалений доступ — для роботи з дому. Нижче наведений перелік моделей доступу.

1. Клієнт серверна модель основана клієнт-серверній архітектурі, ця модель взаємодії між двома основними компонентами: клієнтом і сервером, де клієнт відправляє запити на сервер, а сервер обробляє ці запити та відправляє відповіді. Ця архітектура широко використовується в мережах Інтернету, корпоративних мережах, веб-додатках та інформаційних системах.

2. Однорангова P2P (Peer-to-Peer), ця модель передбачає рівноправний обмін даними між пристроями без центрального сервера. Кожен пристрій у мережі одночасно може бути як клієнтом, так і сервером, а користувачі обмінюються файлами безпосередньо. Ця модель використовується під час роботи з торрентами (протокол BitTorrent), де файли передаються частинами від багатьох користувачів, з програмами для обміну файлами (eMule, Gnutella) та рішеннями для блокчейну та криптовалют (наприклад, Bitcoin).

3. Модель «Віддалений доступ» (Remote Access Model), яка дозволяє користувачам віддалено підключатися до ресурсів організації через інтернет або VPN (віртуальну приватну мережу). Доступ через VPN забезпечує безпечний зашифрований доступ до ресурсів мережі, а RDP (Remote Desktop Protocol) дозволяє користувачу підключатися до робочого столу комп'ютера або сервера з захистом через шифрування TLS або IPsec для шифрування трафіку. Ця модель використовується для віддаленої робота, щоб підключити співробітників до внутрішніх серверів компанії, для адміністрування серверів і для дистанційної техпідтримки.

4. Модель «Хмарні обчислення» (Cloud Computing Model), якій ресурси доступні через інтернет за допомогою "хмарних" провайдерів. Користувачі отримують доступ до серверів, сховищ, додатків і обчислювальних ресурсів через веб-інтерфейс або спеціальні клієнтські програми. Використовується ця модель підтримки Google Drive для зберігання файлів у хмарі, AWS (Amazon Web Services) інфраструктура для запуску віртуальних серверів та сховищ та Microsoft 365 доступ до офісних програм через браузер або додаток.

5. Гібрідна модель (Hybrid model), це поєднання клієнт-серверної та одноранговоі (P2P) моделей. Частина ресурсів керується централізовано (сервер), а частина — за принципом рівноправного обміну між пристроями, це дає гнучкість та надійність за рахунок розподілення навантаження між сервером і рівноправними вузлами. Використовується для Skype та Zoom дзвінків між клієнтами, де дзвінки здійснюються за одноранговою моделлю, а реєстрація та аутентифікація здійснюються на сервері та хмарних ігри (GeForce NOW, Xbox Cloud Gaming), при цьому сервер обробляє ігровий процес, але клієнт може мати локальну кешовану інформацію.

6. Модель «Мобільний доступ» (Mobil Access Model) з якою користувачі можуть отримати доступ до мережевих ресурсів за допомогою мобільних пристроїв (смартфонів, планшетів), доступ із мобільних пристроїв через інтернет або мобільну мережу (Wi-Fi, 4G/5G), а також захищений доступ через VPN або MDM (Mobile Device Management). Використовується для доступу до корпоративних ресурсів з мобільних пристроїв через VPN і для програм віддаленого доступу (наприклад, TeamViewer, AnyDesk).

4.2. Стандартизація у комп'ютерних мережах

Створення, розвиток та поширення використання комп'ютерних мереж були б неможливими без використання однакових правил передачі та обробки даних - стандартів.

Основними організаціями, які розробляють та сприяють впровадженню стандартів в галузі комп'ютерних мереж є:

– ISO (International Organization for Standardization) міжнародна організація, яка займається розробкою стандартів у різних сферах, включаючи комп'ютерні мережі. Модель OSI (Open Systems Interconnection) була розроблена ISO і є основою для стандартизації мережевих протоколів. ISO/IEC 27001 — стандарт безпеки інформаційних технологій.

– IEC (International Electrotechnical Commission) займається створенням стандартів для електронних та електротехнічних пристроїв, включаючи мережеві компоненти та інфраструктуру.

– IETF (Internet Engineering Task Force) основна організація, яка займається розробкою протоколів і стандартів для Інтернету, таких як TCP/IP, HTTP, IPv6. Основний документ, який регулює стандарти для Інтернету, — це RFC (Request for Comments).

– IEEE (Institute of Electrical and Electronics Engineers) розробляє стандарти для мережевих технологій, зокрема для Ethernet, Wi-Fi та інших локальних мереж. Відомі стандарти: IEEE 802.3 (Ethernet), IEEE 802.11 (Wi-Fi), IEEE 802.15 (Bluetooth).

– ITU (International Telecommunication Union) спеціалізується на глобальних стандартів для телекомунікацій і радіозв'язку, включаючи мережі та інтернет-технології. Основний стандарт, що регулює телекомунікаційні мережі — ITU-T (ITU Telecommunication Standardization Sector).

4.3. Семирівнева модель OSI (Open Systems Interconnection)

Модель OSI це концептуальна модель, розроблена Міжнародною організацією стандартизації (ISO), яка визначає стандарти для комунікаційних функцій в комп'ютерних мережах. Вона складається з семи рівнів, кожен з яких відповідає за певну частину обміну даними між комп'ютерами чи іншими пристроями в мережі.

Модель OSI є важливою для стандартизації комунікацій у мережах, оскільки дозволяє розділяти різні функції в мережі на логічні рівні, що полегшує налагодження та розвиток систем (рис. 34). Кожен рівень моделі OSI виконує певне завдання в процесі передачі даних по мережі та є основою для розробки мережевих протоколів. OSI розділяє комунікаційні функції в мережі на сім рівнів, кожен з яких обслуговує різні частини процесу області взаємодії відкритих систем.

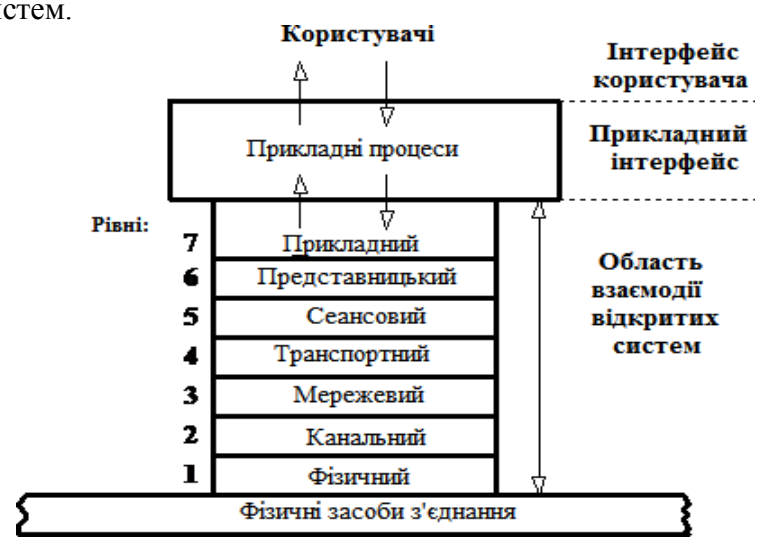

Рисунок 34. Модель OSI

Модель OSI є важливою теоретичною основою для розуміння, проектування та налагодження комп'ютерних мереж. Кожен рівень моделі виконує конкретні функції, і їхня взаємодія забезпечує ефективний і надійний обмін даними між різними пристроями. Модель OSI також слугує основою для розробки стандартів та протоколів, які використовуються в сучасних мережах і надають високий рівень сумісності між різними технологіями.

Модель OSI можна розділити на дві різних моделі (рис. 35), а саме горизонтальну модель на базі протоколів, що забезпечує механізм взаємодії програм і процесів на різних машинах і вертикальну модель на основі послуг, що забезпечуються сусідніми рівнями один до одного на одній машині.

Кожен рівень комп'ютера – відправника взаємодіє з таким же рівнем комп'ютера-одержувача. Такий зв'язок називається логічним або віртуальним так як вони є суміжними рівнями одного комп'ютера. Інформація проходить через усі рівні, а потім передається по фізичному середовищу до комп'ютера-одержувача і знову проходить через всі рівні в зворотному порядку поки не доходить до того ж рівня, з якого вона була послана.

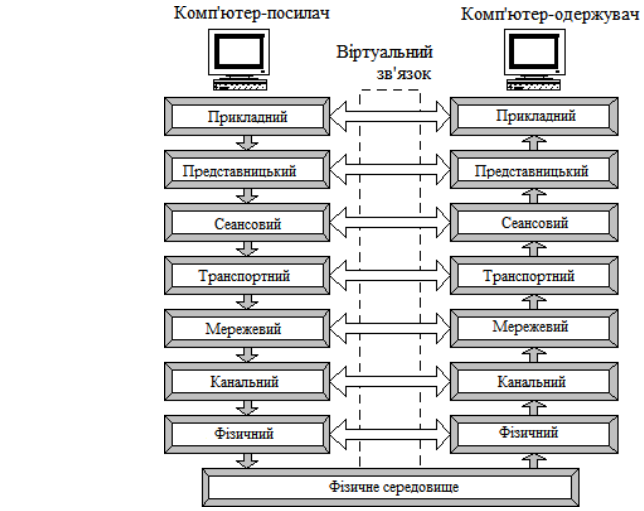

Рисунок 35. Схема взаємодії комп'ютерів у базовій еталонній моделі OSI

Перед поданням в мережу дані розбиваються на пакети. Пакет (packet) – це одиниця інформації, що передається між станціями мережі. При відправці даних пакет проходить послідовно через усі рівні програмного забезпечення. На кожному рівні до пакету додається інформація цього рівня призначена для управління успішною передачею даних по мережі – заголовок (рис. 36), де *Заг* - заголовок пакету, *Кон* - кінець пакету.

На приймаючій стороні пакет проходить через усі рівні в зворотному порядку. На кожному рівні протокол цього рівня читає інформацію із заголовку пакету, яка додана до нього на цьому ж рівні комп'ютера – відправника, потім видаляє її та передає пакет наступному рівню. Коли пакет дійде до *Прикладного* рівня, уся інформація, яка управляє передачею, буде видалена з пакету, а дані наберуть свого первинного вигляду.

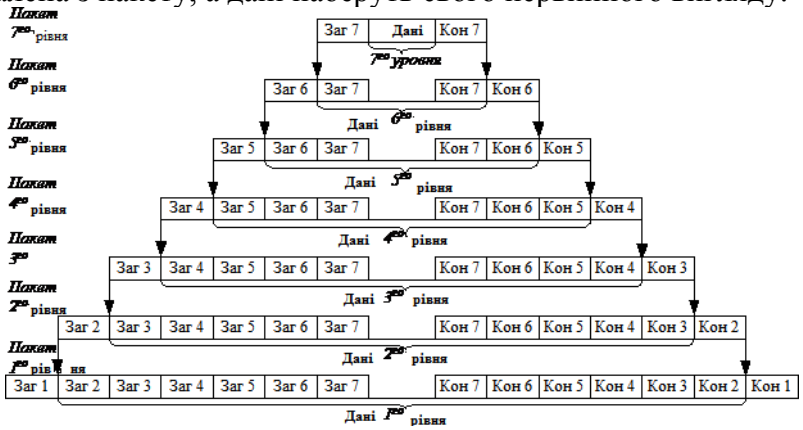

Рисунок 36. Формування пакету кожного рівня семирівневої моделі

Окремі рівні моделі *OSI* зручно розглядати як *групи програм*, призначених для виконання конкретних *функцій*. Один рівень, приміром, відповідає за забезпечення перетворення даних з *ASCII* в *EBCDIC* і містить *програми* необхідні для виконання цього завдання.

Кожен рівень забезпечує сервіс для вище розташованого рівня, посилаючи запит у сервіс нижче розташованого рівня. Верхні рівні в більшості своєї просять сервіс для маршрутизації потрібних даних з однієї мережі в іншу. За практичну реалізацію принципів адресації даних відповідають нижні рівні.

Дана модель визначає взаємодію відкритих систем різних виробників в одній мережі. Тому вона виконує для них координаційні дії з взаємодії прикладних процесів функції: представлення даних, однакового зберігання даних, управління мережевими ресурсами, безпеки даних і захисту інформації, діагностики програм і технічних засобів (рис. 37).

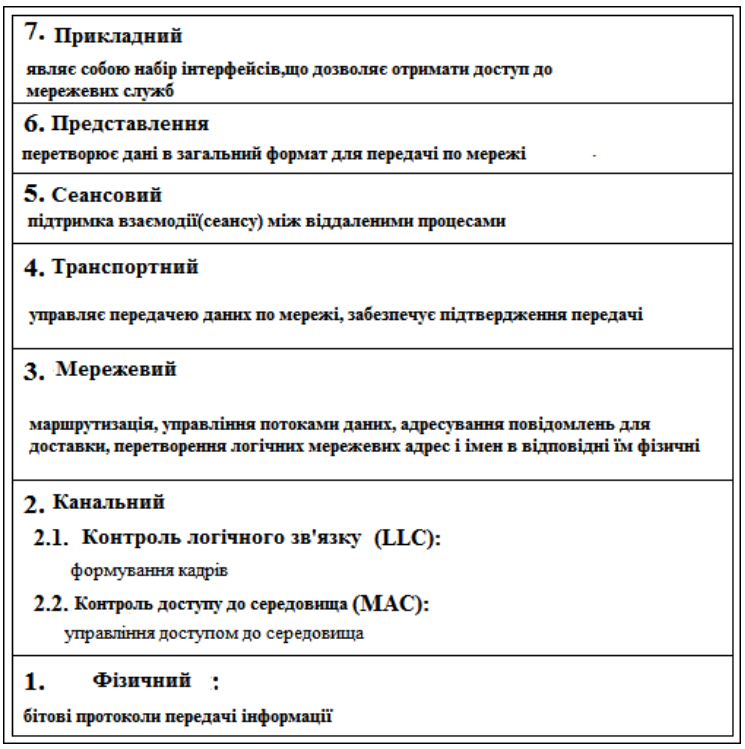

Рисунок 37. Функції рівнів

Прикладний рівень (Application layer) забезпечує прикладним процесам засіб доступу до області взаємодії і безпосередньо примикає до прикладних процесів. На *прикладному* рівні необхідно надати в розпорядження користувачів вже перетворену інформацію, цю задачу бере на себе системне і програмне забезпечення користувача. Цей рівень використовується для взаємодії користувача з мережею, який забезпечує доступ до мережевих сервісів кінцевих користувачів, таких як веб-браузери, поштові клієнти, FTP-сервери та інші мережеві додатки.

Основними завданнями цього рівня є надання користувачам інтерфейсу для доступу до мережі та її ресурсів та забезпечення мережевих послуг для додатків, таких як доступ до баз даних, передача файлів або електронна пошта. Одиниця даних, якою оперує прикладний рівень, називається повідомленням (message).

На цьому рівні використовуються технології HTTP, FTP для веб-браузерів і передачі файлів; SMTP, POP3 і IMAP для електронної пошти; технологія DNS для перетворення доменних імен на IP-адреси.

Рівень представлення даних (Presentation layer) або представницький рівень представляє дані, що передаються між прикладними процесами в потрібній формі.

Цей рівень відповідає за перетворення даних у форму, зрозумілу для додатків, а також за шифрування, стискання, кодування і декодування даних та здійснює перетворення форматів даних, забезпечує їх стандартизацію та сумісність між різними платформами. Крім того в його задачі входить форматування, шифрування і стискування даних для забезпечення сумісності між різними системами.

На цьому рівні використовуються технології JPEG, GIF для зображень; ASCII, EBCDIC для кодування символів; SSL/TLS для шифрування для захисту даних.

Сеансовий рівень (Session layer) – це рівень, що визначає процедуру проведення сеансів між користувачами або прикладними процесами.

Сеансовий рівень відповідає за встановлення, управління і завершення сеансів комунікації між додатками, забезпечує організацію зв’язку між процесами на двох кінцевих системах, а також підтримку діалогу (обмін даними) між ними та організує синхронізацію і відновлення сеансів у разі помилок.

В задачі сеансового рівня входять ідентифікація і контроль сеансів і підтримка діалогу між пристроями.

На цьому рівні використовуються технології NetBIOS для забезпечення мережних операцій вводу/виводу й керування транспортним протоколом нижчого рівня, RPC (Remote Procedure Call) для виклику функцій або процедур в іншому та потокол SMB (Server Message Block), який використовується для надання розділеного доступу до файлів, принтерів і іншим мережевим ресурсам.

Транспортний рівень (Transport Layer) призначений для передачі пакетів через комунікаційну мережу. На транспортному рівні пакети розбиваються на блоки.

Цей рівень відповідає за надійність передачі даних між кінцевими пристроями, виконує сегментацію великих обсягів даних, які надходять із вищих рівнів, і забезпечує доставку цих сегментів, а також забезпечує управління потоком і корекцію помилок на рівні передачі між кінцевими вузлами.

Задачами транспортного рівня є надійна доставка даних між двома кінцевими точками (наприклад, комп'ютерами) і контроль помилок і коригування помилок (наприклад, за допомогою повторних передач).

Технології транспортного рівня визначають протоколи TCP (Transmission Control Protocol), UDP (User Datagram Protocol) і SCTP (Stream Control Transmission Protocol)

Мережевий рівень (Network Layer) зазвичай займається двома основними завданнями це передача даних через фізичні канали зв'язку та маршрутизація даних.

Цей рівень забезпечує маршрутизацію даних між різними мережами, відповідає за логічні адреси (наприклад, IP-адреси) та їх маршрутизацію, регулює передачу даних між різними підмережами та пристроями, а також оптимізує використання маршрутизаторів для вибору найкращого шляху доставки даних.

При організації доставки пакетів на мережевому рівні використовується поняття *номер мережі*. В цьому випадку *адреса* одержувача складається з *номера мережі* і *номера комп'ютера* в цій мережі.

Для організації мережевої інфраструктури використовуються різні пристрої, що виконують функції мережевого рівня:

1. Маршрутизатор (Router), це пристрій, який визначає оптимальний шлях для доставки пакетів даних через різні мережі. Маршрутизатор працює на мережевому рівні, аналізуючи IP-адреси і передаючи пакети між різними підмережами та навіть різними інтернет-підключеннями.

2. Шлюз (Gateway) забезпечує з'єднання між різними мережами, які використовують різні протоколи. Наприклад, шлюз може перетворювати пакети між IP та іншими протоколами, такими як IPX або AppleTalk.

3. Міст (Bridge), це пристрій, який підключає дві або більше мережі на канальному рівні, створюючи одну велику мережу. Мости зазвичай не здійснюють маршрутизацію, але дозволяють з'єднувати різні сегменти локальної мережі.

Мережевий рівень відповідає за ділення користувачів на групи і маршрутизацію пакетів на основі перетворення MAC-адрес в мережеві адреси. Мережевий рівень виконує маршрутизацію даних від джерела до приймача через проміжні пристрої та розбиття даних на пакети для ефективної передачі.

Технології мережевого рівня пов'язані з протоколами IP (Internet Protocol), IPv4, IPv6, протоколами маршрутизації RIP, OSPF, BGP і ICMP для повідомлень про помилки, таких як пінг.

Канальнй рівень (Data Link) відповідає за формування кадрів (data frames) і забезпечує надійну передачу даних між двома пристроями в мережі, за виявлення та корекція помилок, що можуть виникнути при передачі на фізичному рівні та контролює доступ до фізичного середовища.

Кадри – це логічно організована структура, в яку можна поміщати дані. Створення кадрів і додавання до них необхідної інформації для управління передачею одна з основних

задач канального рівня. Крім того окремі функції включають адресацію на рівні MAC (Media Access Control) та визначення колізій в мережах, що використовують спільне середовище передачі.

Канальний рівень забезпечує коректність передачі кожного кадру, поміщаючи спеціальну послідовність біт, в початок і кінець кожного кадру, щоб відмітити його, а також обчислює певним способом контрольну суму, яка дорівнює сумі усіх байтів кадру, і додаючи контрольну суму до кадру. Коли кадр приходить, одержувач знову обчислює контрольну суму отриманих даних і порівнює результат з контрольною сумою з кадру, якщо вони співпадають, кадр вважається правильним і приймається. Якщо ж контрольні суми не співпадають, то фіксується помилка.

Канальний рівень забезпечує створення, передачу і прийом кадрів даних. Цей рівень обслуговує запити мережевого рівня і використовує сервіс фізичного рівня для прийому і передачі пакетів. Специфікації IEEE 802.X ділять канальний рівень на два підрівні:

LLC (*Logical Link Control*) управління логічним каналом здійснює логічний контроль зв'язку. Підрівень LLC забезпечує обслуговування мережевого рівня і пов'язаний з передачею і прийомом призначених для користувача повідомлень.

MAC (*Media Assess Control)* контроль доступу до середовища. Підрівень MAC регулює доступ до розділеного фізичного середовища (передача маркера або виявлення колізій або зіткнень) і управляє доступом до каналу зв'язку. Підрівень LLC знаходиться вище за підрівень *MAC.*

Канальний рівень використовує технологію Ethernet для локальних мереж і протоколи PPP (Point-to-Point Protocol) й Wi-Fi (802.11).

Фізичний рівень (Physical Layer) призначений для сполучення з *фізичними засобами з'єднання*. *Фізичні засоби з'єднання* – це сукупність *фізичного середовища,* апаратних і програмних засобів, що забезпечує передачу сигналів між системами. *Фізичне середовище* – це матеріальна субстанція, через яку здійснюється передача сигналів. Фізичне середовище є основою, на якій будуються фізичні засоби з'єднання. Як фізичне середовище широко використовуються ефір, метали, оптичне скловолокно і кварц.

Фізичний рівень визначає фізичні характеристики передачі даних через комунікаційний канал, описує апаратне забезпечення, що забезпечує передачу бітів, наприклад, кабелі, роз'єми, електричні сигнали та забезпечує механізм для передачі бітів через фізичні середовища (кабелі, оптоволокно, бездротові канали).

Фізичний рівень виконує задачі визначення механізмів кодування та модуляції сигналів, передача бітів (0 та 1) по каналах зв'язку і підтримка фізичних з'єднань між пристроями.

До технологій цього рівня відносяться Ethernet (кабельні з'єднання), Wi-Fi (бездротові з'єднання), Bluetooth і USB (для підключення периферійних пристроїв)

4.4. Протоколи і стеки протоколів.

Погоджений набір протоколів різних рівнів, достатній для організації міжмережевої взаємодії, називається *стеком протоколів* [12]. Для кожного рівня визначається набір функцій – запитів для взаємодії з рівнем, що лежить вище, який називається *інтерфейсом*. Правила взаємодії двох машин можуть бути описані у вигляді набору процедур для кожного з рівнів, які називаються *протоколами*.

Існує досить багато стеків протоколів, широко вживаних в мережах. Це і стеки, що є міжнародними і національними стандартами, і фірмові стеки, що отримали поширення завдяки поширеності устаткування тієї або іншої фірми. Прикладами популярних стеків протоколів можуть служити стек IPX/SPX фірми Novell, стек TCP/IP, використовуваний в мережі Internet і у багатьох мережах на основі операційної системи UNIX, стек OSI міжнародної організації по стандартизації, стек DECnet корпорації Digital Equipment і деякі інші.

Стеки протоколів розбиваються на три рівні:

– мережеві;
– транспортні;
– прикладні.

Мережеві протоколи надають наступні послуги: адресацію і маршрутизацію інформації, перевірку на наявність помилок, запит повторної передачі і встановлення правил взаємодії в конкретному мережевому середовищі. Нижче наведено найбільш популярні мережеві протоколи.

DDP (Datagram Delivery Protocol) – протокол доставки дейтаграмм. Протокол передачі даних Apple, який використовується в Apple Talk.

IP (Internet Protocol) – протокол стека TCP/IP, що забезпечує адресну інформацію і інформацію про маршрутизацію.

IPX (Internetwork Packet eXchange) – міжмережевий обмін пакетами в NWLink. Протокол Novel NetWare, який використовується для маршрутизації.

NetBEUI (NetBIOS Extended User Interface) – розширений, призначений для користувача інтерфейс базової мережевої системи введення виведення. Розроблений спільно IBM і Microsoft, цей протокол забезпечує транспортні послуги для *NetBIOS.*

Транспортні протоколи надають наступні послуги надійного транспортування даних між комп'ютерами. Нижче наведено найбільш популярні транспортні протоколи.

ATP (Apple Talk Protocol) – протокол транзакцій Apple Talk і NBP (Name Binding Protocol) – протокол зв'язування імен. Сеансовий і транспортний протоколи Apple Talk.

*NetBIOS – б*азова мережева система введення/виведення. NetBIOS встановлює з'єднання між комп'ютерами, а NetBEUI надає послуги передачі даних для цього з'єднання.

SPX (Sequenced Packet eXchange) – послідовний обмін пакетами в NWLink.

Novel NetWare – протокол, що використовується для забезпечення доставки даних.

TCP (Transmission Control Protocol) – протокол управління передачею. Протокол стека TCP/IP, що відповідає за надійну доставку даних.

Прикладні протоколи відповідають за взаємодію застосувань. Нижче наведено найбільш популярні прикладні протоколи.

AFP (Apple Talk File Protocol) – файловий протокол Apple Talk для віддаленого управління файлами Macintosh.

FTP (File Transfer Protocol) – протокол передачі файлів стека TCP/IP.

NCP (NetWare Core Protocol) – базовий протокол NetWare. Оболонка і редиректори клієнта Novel NetWare.

SNMP (Simple Network Management Protocol) – простий протокол управління мережею і спостереження за мережевими пристроями стека TCP/IP.

HTTP (Hyper Text Transfer Protocol) – протокол передачі гіпертексту і інші протоколи.

Для налаштування локальної мережі у Windows за допомогою роутера у формі властивостей системи необхідно «Змінити параметри» ім`я комп`ютера і домену, до якого належить даний вузол. У вкладці «Ім'я комп'ютера» крім цього можна змінити робочу групу, за замовчанням стоїть "WORKGROUP". Існує альтернативний спосіб потрапити на ту ж саму форму за допомогою послідовності клавіш "Win + R" і ввести команду sysdm.cpl. Налаштувати додаткові параметри можна в центрі управління загальним доступом і мережевими підключеннями, там можна внести зміни в опцію «Інтернет і підключення до мережі» вкладки «Перегляд стану завдань і мережі». З параметрами налаштування захисту мережі можна познайомитись за адресою https://support.microsoft.com.

ЗАВДАННЯ

1 Перевірте IP-адресу, маску підмережі, шлюз за замовчуванням, Mac-адресу та доменну адресу комп'ютера.

2 Налаштуйте локальну мережу, змініть ім'я комп'ютера і робочу групу Налаштуйте параметри захисту.

КОНТРОЛЬНІ ПИТАННЯ

1. За якими класифікаційними ознаками визначаються комп’ютерні мережі?
2. Що таке топологія?
3. Перерахуйте найбільш використовувані типи топології мережі?
4. Які технологій відносяться до мережевих?
5. Що таке моделі доступу до ресурсів, чим вони відрізняються?
6. Перерахуйте комп’ютерні мережі за розміром.
7. Якими ресурсами оперує комп’ютерна мережа?
8. Для чого потрібні мережеві стандарти?
9. Які функції виконує модель OSI?
10. Кратко опишіть функції рівнів моделі OSI.
11. Для чого призначені мережеві протоколи?
12. Що таке метод доступу і як впливає метод доступу на передачу даних в мережі?
13. Що таке стек протоколів?

СПИСОК ЛІТЕРАТУРИ

1. Tanenbaum Andrew S. Computer architektur – Strukturen, Konzepte, Grundlagen, 6th Edition. Pearson Education, 2016. – 829 p.
2. Segal R. Space Packed: The Architecture of Alfred Neumann. Zurich : Park Books, 2018. 352 p.
3. Кравчук С. О., Шохін В. О. Основи комп'ютерної техніки: Компоненти, системи, мережі : навчальний посібник. Київ: Каравела, 2006. 344 с.
4. Тонкошкур О. С., Гниленко О. Б., Матвєєва Н. О., Морозов О. С. Архітектура комп'ютерів. Машинні команди та програмування на асемблері: навчальний посібник. Дніпро : Вид-во «Нова Ідеологія», 2018. 179 с.
5. Валецька Т. М. Апаратні засоби персональних комп'ютерів: навчальний посібник. Київ: Центр навчальної літератури, 2002. 208 с.
6. Тарарака В. Д. Архітектура комп'ютерних систем: навчальний посібник. Житомир : ЖДТУ, 2018. 383 с.
7. Матвієнко М. П., Розен В. П., Закладний О. М. Архітектура комп'ютера. Навчальний посібник. Київ : Видавництво Ліра-К. 2016. 256 с.
8. Мельник А. О. Архітектура комп'ютера. – Луцьк: Волинська обласна друкарня, 2008. 470 с.
9. Кулаков Ю.О., Жуков І. А. Комп'ютерні мережі : навч. посібник з грифом МОН України Вид-во Нац. Авіа. університет «НАУ-друк», 2009. 329с.
10. Микитишин А. Г., Митник М. М., Стухляк П. Д., Пасічник В. В. Комп'ютерні мережі. Книга 1 : навч. посібник. – Львів : «Магнолія 2006», 2013. 256 с.
11. Микитишин А. Г., Митник М. М., Стухляк П. Д., Пасічник В. В. Комп'ютерні мережі. Книга 2 : навч. посібник. – Львів : «Магнолія 2006», 2013. 328 с.
12. ISO/IEC 27001:2022(en) Information security, cybersecurity and privacy protection — Information security management systems — Requirements
13. Допомога та навчання по Windows інтернет ресурс: https://support.microsoft.com/windows

Навчальне видання

ГРИГОРОВА ТЕТЯНА АЛЬБЕРТІВНА

АРХІТЕКТУРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

Електронне видання

Формат. Гарнітура Times New Roman.
Ум. друк. арк. . Обл.-вид. арк. .