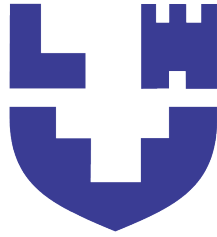


**Міністерство освіти і науки України
Луцький національний технічний університет**



АРХІТЕКТУРА КОМП'ЮТЕРІВ

Конспект лекцій
для здобувачів першого (бакалаврського) рівня вищої освіти
освітньої програми «Комп'ютерна інженерія»
галузі знань 12 «Інформаційні технології»
спеціальності 123 «Комп'ютерна інженерія»
денної та заочної форм навчання

Луцьк 2025

УДК 004.451

A-87

Рекомендовано до видання вченою радою факультету комп'ютерних та інформаційних технологій ЛНТУ, протокол № _____ від «___» _____ 2025 року

Голова вченої ради факультету КІТ _____ Інна КОНДІУС

Електронна копія друкованого видання передана для внесення в репозитарій ЛНТУ
Директор бібліотеки _____ Наталія ПОЛІЩУК

Розглянуто і схвалено на засіданні кафедри комп'ютерної інженерії та безпеки ЛНТУ, протокол № _____ від «___» _____ 2025 року

Укладач: _____ Наталія ХРИСТИНЕЦЬ, кандидат технічних наук, кафедри комп'ютерної інженерії та безпеки ЛНТУ

Рецензент: _____ Наталія ЯКИМЧУК, кандидат технічних наук, доцент кафедри електроніки та телекомунікацій ЛНТУ

Відповідальний за випуск: _____ Тарас ТЕРЛЕЦЬКИЙ, кандидат технічних наук, доцент, завідувач кафедри комп'ютерної інженерії та безпеки ЛНТУ

A-87

Архітектура комп'ютерів: конспект лекцій для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерна інженерія» галузі знань 12 «Інформаційні технології» спеціальності 123 Комп'ютерна інженерія денної та заочної форм навчання / уклад. Н.А. Христинець. Луцьк : ЛНТУ, 2025. 112 с.

Конспект лекцій складено відповідно до діючої програми курсу «Архітектура комп'ютерів». Призначений для здобувачів вищої освіти спеціальності 123 Комп'ютерна інженерія освітньої програми «Комп'ютерна інженерія». Може бути використаний студентами технічних спеціальностей при вивченні даної дисципліни.

Н.А. Христинець 2025

ЗМІСТ

ТЕМА 1. ОСНОВНІ ЕТАПИ РОЗВИТКУ КОМП'ЮТЕРІВ	4
ТЕМА 2 ФУНКЦІЇ, СТРУКТУРА ТА ХАРАКТЕРИСТИКИ КОМП'ЮТЕРА	18
ТЕМА 3. ТИПИ АРХІТЕКТУР КОМП'ЮТЕРА	26
ТЕМА 4 ПАМ'ЯТЬ ТА ІЄРАРХІЯ ПАМ'ЯТІ	43
ТЕМА 5. БУДОВА ПРОЦЕСОРА І РЕГІСТРОВОЇ ПАМ'ЯТІ.....	52
ТЕМА 6. ФУНКЦІЇ ТА ЕЛЕМЕНТАРНІ ОПЕРАЦІЇ АРИФМЕТИКО- ЛОГІЧНОГО ПРИСТРОЮ	60
ТЕМА 7. ПРИСТРІЙ КЕРУВАННЯ У СКЛАДІ ПРОЦЕСОРА	66
ТЕМА 8. КОДУВАННЯ КОМАНД В КОМП'ЮТЕРІ.....	71
ТЕМА 9. СИСТЕМИ КОМАНД КОМП'ЮТЕРА. АРХІТЕКТУРИ CISC ТА RISC	78
ТЕМА 10. КОНВЕЄРНЕ ВИКОНАННЯ КОМАНД	85
ТЕМА 11. СПОСОБИ АДРЕСАЦІЇ ОПЕРАНДІВ	89
ТЕМА 12. ОРГАНІЗАЦІЯ ВВЕДЕННЯ-ВИВЕДЕННЯ.....	94
ТЕМА 13. ПАРАЛЕЛІЗМ ТА БАГАТОПОТОЧНІСТЬ.....	99
ТЕМА 14. АРХІТЕКТУРИ КЛАСТЕРІВ ТА СУПЕРКОМП'ЮТЕРІВ.....	104
СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ.....	110

ТЕМА 1. ОСНОВНІ ЕТАПИ РОЗВИТКУ КОМП'ЮТЕРІВ

Історія компанії IBM та відомих виробників процесорної техніки: Intel Corporation, Advanced Micro Devices (AMD), Apple Inc, ARM Holdings, Qualcomm Incorporated. Сучасні тенденції розвитку архітектури комп'ютерів: x86 та ARM. Квантовий комп'ютер. Оптичні комп'ютери. Фотоний комп'ютер. Біокомп'ютинг.

Історія розвитку комп'ютерної техніки відзначається безперервним прогресом від примітивних механічних обчислювальних пристроїв до складних електронних систем, що стали невід'ємною частиною сучасного життя. Вона охоплює кілька ключових етапів, кожен з яких приніс нові технології та інновації [1].

Ручні та механічні засоби обчислень раннього періоду.

Серед перших пристроїв для полегшення обчислень можна виділити спеціальну дошку, відому як абак, що використовувалася вже у V-IV ст. до н.е. Обчислення на абакі проводилися шляхом переміщення кісток або камінців в поглибленнях дошки, виготовленої з бронзи, каменю, слонової кістки тощо. В Греції абак був відомий вже в V ст. до н.е., у Японії його називали «серобаян», а в Китаї - «суанпан» (рис. 1.1).



Рисунок 1.1 – Древній суансап як прототип сучасного комп'ютера

У 1642 році французький математик і філософ Блез Паскаль створив першу обчислювальну машину, що виконувала операцію додавання, відому як Паскалін. Цей механічний пристрій у вигляді ящика з численними шестернями міг не тільки складати, але й віднімати числа. У 1673 році Готфрід Вільгельм Лейбніц створив механічний обчислювальний пристрій, відомий як ступінчастий обчислювач Лейбніца або Колесо Лейбніца. Англійський математик Чарльз Беббідж розробив пристрій, який не тільки виконував арифметичні операції, але й відразу друкував результати. У 1832 році була виготовлена десятикратно зменшена модель з двох тисяч латунних деталей, що важила три тонни. Ця модель була здатна виконувати арифметичні операції

з точністю до шостого знака після коми і обчислювати похідні другого порядку. Ця обчислювальна машина, відома як диференціальна машина, стала прообразом сучасних комп'ютерів. У 1936 році молодий вчений з Кембриджу Алан Тьюринг представив уявний обчислювальний апарат, що існував лише на папері. Його «розумна машина» працювала за певним алгоритмом, і, в залежності від алгоритму, могла виконувати різноманітні обчислювальні завдання. Хоча на той час це були лише теоретичні концепції, ці ідеї стали основою для створення програмованих комп'ютерів, що обробляють дані відповідно до заданих команд.

Інформаційні революції в історії

В історії людської цивілізації можна виділити кілька ключових інформаційних революцій, що значно вплинули на соціальні структури та відносини завдяки змінам в обробці, збереженні та передачі інформації.

Перша революція відбулася з винаходом писемності, що радикально змінило спосіб передачі знань від одного покоління до іншого. Це дозволило людству зберігати та передавати інформацію в письмовій формі, створюючи базу для подальшого культурного та наукового розвитку.

Друга революція, що відзначилася у середині XVI століття, була спричинена винаходом друкарства. Це нововведення кардинально змінило індустриальне суспільство, вплинуло на культуру та організацію діяльності, забезпечивши масове розповсюдження знань і інформації.

Третя революція відбулася наприкінці XIX століття, коли були відкриті нові можливості завдяки електриці. Виникнення телеграфу, телефону та радіо забезпечило швидку передачу та накопичення інформації у великих обсягах, що суттєво вплинуло на комунікацію та соціальні процеси.

Четверта революція, що розпочалася у 70-х роках XX століття, була пов'язана з появою мікропроцесорної технології та персональних комп'ютерів. Це період характеризується трьома основними інноваціями:

- перехід від механічних та електричних засобів обробки інформації до електронних технологій;
- мініатюризація всіх компонентів, пристроїв та систем, що дозволило створити компактні, але потужні комп'ютерні системи;
- розвиток програмно-керованих пристроїв та процесів, що дало можливість автоматизувати обробку даних і зменшити залежність від людського втручання.

Історія розвитку комп'ютерної техніки

Потреба в зберіганні, перетворенні та передачі інформації у людей виникла значно раніше, ніж були створені телеграф, перша телефонна станція та електронна обчислювальна машина (ЕОМ). Весь накопичений людством досвід і знання стали основою для розвитку обчислювальної техніки. Історія створення ЕОМ, що є загальною назвою для електронних машин, призначених для обчислень, бере свій початок у далекому минулому та тісно пов'язана з різними аспектами людської діяльності. Від початку цивілізації існували різні

форми автоматизації обчислень. Розвиток комп'ютерної техніки триває вже близько п'яти десятиліть, протягом яких змінювались кілька поколінь ЕОМ. Кожне нове покоління відрізнялося новими елементами (електронні лампи, транзистори, інтегральні схеми) та принципово іншими технологіями виготовлення.

Наразі прийнято виділяти такі покоління ЕОМ:

- 1-е покоління (1946 - початок 50-х років): використовувалися електронні лампи. Ці ЕОМ були великих розмірів, споживали багато енергії, мали низьку швидкість і надійність, а програмування здійснювалося в кодах;

- 2-е покоління (кінець 50-х - початок 60-х років): використовували напівпровідникові елементи, що дозволило покращити технічні характеристики. Програмування стало можливим за допомогою алгоритмічних мов;

- 3-е покоління (кінець 60-х - кінець 70-х років): інтегральні схеми і багат шаровий друкований монтаж. Це забезпечило зменшення розмірів ЕОМ, підвищення надійності і продуктивності, а також доступ до віддалених терміналів;

- 4-е покоління (з середини 70-х - кінець 80-х років): мікропроцесори і великі інтегральні схеми;

- 5-е покоління (з середини 80-х років): розпочалася розробка інтелектуальних комп'ютерів, розширилось впровадження комп'ютерних мереж, розподіленої обробки даних та комп'ютерних інформаційних технологій.

Зі зміною поколінь змінювався і характер використання. Спочатку комп'ютери призначалися переважно для обчислювальних завдань, але згодом їх застосування значно розширилося: обробка інформації, автоматизація управлінських і наукових процесів тощо.

Історія компанії IBM та відомих виробників процесорної техніки: Intel Corporation, Advanced Micro Devices (AMD), Apple Inc, ARM Holdings, Qualcomm Incorporated

Техногігант IBM – це одна з найстаріших та найуспішніших компаній у світі. Її акції мають стійкий попит у інвесторів, оскільки IBM демонструє стабільний розвиток і високі фінансові показники. IBM працює на ринку ІТ вже понад 100 років і постійно інвестує у дослідження та розробки, створюючи нові технології, які змінюють світ. Компанія відома створенням безлічі технологічних інновацій, у тому числі карт з магнітною смужкою, дискет, першого персонального комп'ютера [2]

Intel Corporation, або просто Intel, є одним з найбільших виробників комп'ютерних мікропроцесорів у світі. Компанія була заснована в 1968 році та швидко стала лідером у галузі мікропроцесорів, домінуючи на ринку протягом багатьох років. Intel – компанія світового рівня, яка виготовляє процесори та мікрочіпи для серверів та комп'ютерів. Сьогодні про Intel знає

кожен, хто якимось пов'язаний з даною сферою, але спочатку це був тільки стартап двох бізнесменів-початківців.

Історія компанії Intel починається, коли Роберт Нойс та Гордон Мур заснували компанію з метою виробництва інтегральних схем. У той час вони працювали в компанії Fairchild Semiconductor, де були винахідниками першої комерційної інтегральної схеми. За короткий час після заснування Intel стали виробляти більш потужні мікропроцесори, які змінили світ комп'ютерів.

Історія цієї компанії сягає від випуску Intel 4004 – першого у світі комерційного мікропроцесора, який став революцією в обчислювальній техніці в 1971 році – до 2022 року: випуску процесорів 13-го покоління Intel Core (Raptor Lake), які подальше розвивають гібридну архітектуру з підвищеною продуктивністю та енергоефективністю та 2024 рік: випуск нових продуктів для центрів обробки даних та продовження роботи над мікроархітектурою Meteor Lake.

Назва компанії – це об'єднане словосполучення **Integrated Electronics**.

Один із засновників компанії Г. Мур ще до появи Інтел висунув теорію, за якою кожні 18 місяців системи обчислення будуть складнішими у 2 рази. Процесори стануть дешевшими і швидше. Це твердження навіть стали називати законом Мура. (спростування після Nehalem)

Старт компанії Інтел відбувався з випуску електронних годинників. Це був дуже модний аксесуар на той час. Але компанія не витримала конкуренції на той час і продала бренд годинника.

Бізнес-план, який став основою запуску бізнесу на початку, займав лише 1 сторінку друкованого тексту на машинці.

Процесор Інтел – це кремній, який виготовляють із морського піску. Сьогодні Intel має навіть музей, який дуже люблять відвідувачі з усього світу. Він розташований за адресою: 2200 Mission College Blvd, Santa Clara, CA 95054, Сполучені Штати. Вхід безкоштовний. Музей відкрили ще у 1992 року. Площа музею близько 1000 кв. Виробництво Intel працює цілодобово (як тех. підтримка хостинг-компанії ГіперХост).

Площа виробництв (рис. 1.4) компанії Intel сягає понад мільйон квадратних метрів. Це робить їх одними з найбільших виробничих потужностей у світі. Розташування таких заводів по всьому світу обрано таким чином, щоб оптимізувати виробництво та логістику, а також забезпечити доступ до висококваліфікованої робочої сили та найсучасніших технологічних центрів. Заводи Intel, зокрема у США, Ірландії та Азії, виготовляють мікропроцесори за найсучаснішими технологіями, включаючи виготовлення чіпів на основі 5-нанометрових та навіть 3-нанометрових технологічних процесів. Це дозволяє Intel залишатися лідером у виробництві передових напівпровідників для персональних комп'ютерів, серверів та мобільних пристроїв.



Рисунок 1.2 – Виробничі потужності Intel Co.

Цікавим аспектом є високий рівень автоматизації та інновацій, які використовуються на заводах Intel. Процес виробництва чіпів є надзвичайно складним і потребує високоточних технологій для створення мікросхем з мільярдами транзисторів. Для цього використовуються спеціалізовані фабрики, оснащені найновішими технологіями, такими як фотолітографія, іонне легування та складання чіпів на нанорівні. Крім того, Intel активно інвестує в модернізацію своїх заводів, що дозволяє не тільки виробляти продукцію високої якості, але й зменшувати вплив виробничих процесів на навколишнє середовище.

У 1974 на цих виробництвах був розроблений перший 8 бітний мікропроцесор 8080, пізніше на його честь був названий астероїд 8080 Intel.

Офіси Intel - це повністю опен-спейс, навіть керівництво не сидить у кабінетах. Це створено для того, щоб співробітники компанії могли безперервно спілкуватися. Щороку компанії бере участь у благодійних акціях та витрачає понад 100 млн доларів. Наприклад, Intel часто спонсорує освітні проекти. Наприклад, завод Intel в Ірландії розташований у місті Лейклейш (Leixlip), де компанія має одну з найбільших своїх фабрик у Європі (рис.1.3).

Цей завод був відкритий у 1989 році і став важливим центром виробництва мікропроцесорів. Завод спеціалізується на виготовленні процесорів для різних застосувань, від персональних комп'ютерів до серверів і мобільних пристроїв. Intel інвестує в розвиток виробничих потужностей і вдосконалення технологій, щоб зберігати свою лідируючу позицію на ринку мікропроцесорів.



Рисунок 1.3 – Завод Intel в Ірландії

У компанії Інтел один із логотипів для процесора був не зовсім вдалим. Йдеться про логотип для процесора 386. Це аналог колишнього логотипу з цифрою 286, яка перекреслена червоним хрестом. Користувачі не оцінили таку оригінальність, і компанія відмовилася від використання лого. Перекреслення символізувало відмову від старих процесорів.

AMD – це американська технологічна компанія, що виробляє процесори, графічні процесори та інші компоненти комп'ютерів. Компанія була заснована в 1969 році і відома своїми інноваціями та конкурентоспроможними продуктами.

Компанія була заснована Джеррі Сандерсом, який раніше працював в компанії Fairchild Semiconductor. У 1969 році він зібрав команду з 7 інженерів і заснував компанію Advanced Micro Devices (AMD) в Санта-Кларі, Каліфорнія. Початково компанія спеціалізувалася на виробництві логічних елементів, що використовуються в електронних пристроях. У 1980 році компанія AMD випустила свій перший 16-бітний мікропроцесор Intel-сумісний, AMD Am9080. Однак цей продукт не знайшов широкого застосування, і компанія продовжувала працювати над удосконаленням своїх продуктів. В 1991 році AMD випустила свій перший 386-сумісний процесор, AMD 386DX-40. Він був конкурентним з продуктами Intel, але був доступний за більш низькою ціною. Він став дуже популярним у виробників комп'ютерів, які хотіли заощадити на компонентах. У 2003 році AMD випустила свій перший 64-бітний процесор AMD Opteron. Цей процесор був спроектований для серверів та робочих станцій і став першим комерційно доступним 64-

бітним процесором. Він був конкурентним з продуктами Intel, але мав кращі показники продуктивності. У 2006 році AMD придбала компанію ATI Technologies, що виробляє графічні процесори. Це дало AMD можливість конкурувати з Intel та Nvidia. У 2011 році AMD випустила свій перший 8-ядерний процесор AMD FX (рис. 1.5). Цей процесор був спроектований для геймерів та робочих станцій, і він був конкурентним з продуктами Intel. Він мав високу продуктивність і став дуже популярним серед геймерів та професійних користувачів.



Рисунок 1.5 – Процесор Ryzen 7 AMD

У 2017 році AMD випустила свою лінійку процесорів AMD Ryzen, яка змінила картину на ринку комп'ютерних процесорів. Процесори Ryzen мали високу продуктивність та були доступні за більш низькою ціною, ніж продукти Intel. Лінійка Ryzen стала дуже популярною серед геймерів та професійних користувачів.

У 2019 році AMD випустила свої перші 7-нм процесори, які були конкурентними з продуктами Intel. Ці процесори мали високу продуктивність і були доступні за більш низькою ціною, ніж продукти Intel. Це змінило картину на ринку комп'ютерних процесорів та підвищило конкуренцію між AMD та Intel.

У сучасному світі AMD є однією з провідних компаній у галузі виробництва компонентів комп'ютерів. Компанія продовжує розробляти нові продукти та інноваційні технології, які допомагають їй змагатися з Intel та іншими провідними компаніями у галузі технологій.

ARM Holdings (колишня аббревіатура Advanced RISC Machines) – британська корпорація, один з найбільших розробників і ліцензіарів архітектури RISC-процесорів (ARM), орієнтованих на використання в портативних і мобільних пристроях (телефонах, органайзерах тощо).

Arm Holdings не є виробником мікропроцесорів, однак ліцензує власну технологію іншим компаніям-виробникам: AMD, Atmel, Cirrus Logic, Intel,

Marvell, NXP, Samsung, Qualcomm, MediaTek, Sony Ericsson, Texas Instruments, які, власне, і втілюють її в чипах.

Технологія ARM виявилася дуже успішною і є домінуючою мікропроцесорною архітектурою для портативних цифрових пристроїв. Arm стверджує, що загальний обсяг мікропроцесорів, зроблених за ліцензіями, перевищує 2,5 мільярдів штук.

Окрім технологій CPU, Arm також розробляє та ліцензує графічні процесори Mali, які найчастіше зустрічаються в Android-планшетах та іноді в смартфонах. В графічному сегменті Arm конкурує з Imagination Technologies PowerVR, Qualcomm Adreno, Nvidia та розробками Intel (угода з поглинання компанії Nvidia, оголошена в 2020 році, провалилася в лютому 2022 року).

Nvidia Corporation спеціалізується на виробництві процесорів для графічних карт та інших візуальних пристроїв. Їх процесори зазвичай мають високу продуктивність та дозволяють виконувати складні завдання, пов'язані з обробкою зображень та відео. Nvidia також випускає процесори для обчислювальних задач, таких як штучний інтелект, машинне навчання та глибоке навчання.

Apple Inc. є виробником процесорів для своїх власних пристроїв, таких як iPhone, iPad та Mac. Вони використовують свою власну технологію процесорів, відому як Apple Silicon, яка зазвичай має високу продуктивність та енергоефективність. Процесори Apple Silicon також дозволяють додатком, розробленим для MacOS, працювати на пристроях з чіпами M1-M4 без необхідності перетворення для архітектури Intel.

Qualcomm Incorporated є головним виробником процесорів для мобільних пристроїв, таких як смартфони та планшети. Їх процесори зазвичай мають високу продуктивність та енергоефективність, що дозволяє забезпечити тривалу роботу батареї на мобільних пристроях. Вони також спеціалізуються на виробництві процесорів для Інтернету речей та інших бездротових пристроїв.

Усі виробники процесорів грають важливу роль у технологічній індустрії, забезпечуючи нам можливість виконувати різні завдання з різною складністю. Кожен з них має свої унікальні технології та підходи до розробки процесорів, що дає можливість обирати продукти залежно від потреб користувачів. Очікується, що з розвитком технологій виробники процесорів будуть продовжувати робити вагомий внесок в електронну індустрію та прискорювати її розвиток.

Сучасні тенденції розвитку архітектури комп'ютерів

Архітектура комп'ютера охоплює широке коло проблем, пов'язаних з побудовою комплексу апаратних і програмних засобів і враховують безліч чинників. Серед цих компонентів найважливішими є: вартість, сфера застосування, функціональні можливості, зручність експлуатації, енергоефективність.

Основні компоненти архітектури комп'ютера можна представити у вигляді схеми, показаної на рисунку 1.6.

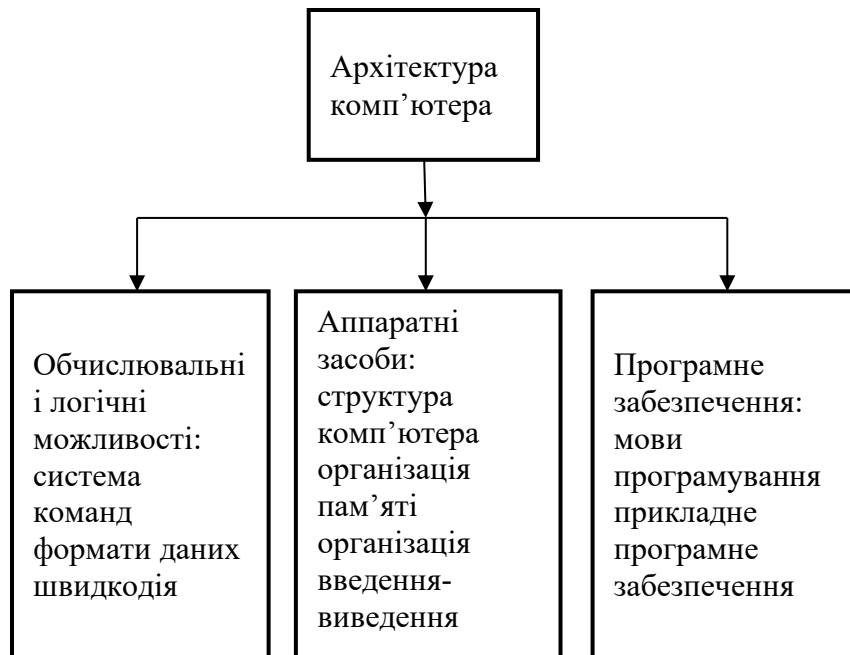


Рисунок 1.6 – Основні компоненти архітектури комп'ютера

Архітектуру обчислювального засобу слід відрізнити від його структури. Структура обчислювального засобу визначає його конкретний склад на деякому рівні деталізації (пристрої, блоки, вузли і т.д.) і описує зв'язки всередині кошти у всій його повноті. Архітектура ж визначає правила взаємодії складових частин обчислювального засобу, опис яких виконується в тій мірі, в якій це необхідно для формування правил їх взаємодії. Вона регламентує не всі зв'язки, а найбільш важливі, які повинні бути відомі для найбільш грамотного використання даного засобу.

Так користувачеві комп'ютера байдуже, на яких елементах виконані електронні схеми, як реалізовані команди і т.д. Важливо інше: які можливості надаються користувачеві, як ці можливості пов'язані зі структурними особливостями комп'ютера, як пов'язані між собою характеристики окремих пристроїв, що входять до складу комп'ютера, і який вплив вони чинять на загальні характеристики машини. Іншими словами, архітектура комп'ютера дійсно відображає коло питань, що відносяться до загального проектування і побудови обчислювальних машин та їх програмного забезпечення.

Таким чином, по ходу вивчення предмета ми, по можливості, розглянемо всі компоненти комп'ютера, включаючи структуру комп'ютера, організацію пам'яті, організацію введення-виведення даних, принципи управління комп'ютера, системи команд, склад програмного забезпечення.

Аналіз сфер використання комп'ютерів показує, що можна виділити два основних напрямки їх використання. **Перший напрямок** – це поліпшення інтелектуальних можливостей людини. Мається на увазі прискорення обчислень, зберігання великих об'ємів інформації, швидкий пошук та відображення необхідної інформації і т. д. Комп'ютери цього напрямку повинні мати велику ємність пам'яті, потужне програмне забезпечення, розвинуті засоби взаємодії з людиною. **Другий напрямок** – це використання

комп'ютера як системи керування, інформаційно-вимірювальної системи, системи передачі даних і т. д. Основна вимога до комп'ютерів даного напрямку – забезпечення необхідного часу реакції на події в системі, проведення обробки вхідних даних в темпі їх надходження, тобто в реальному масштабі часу, і видача результатів відповідно до часових вимог системи. Комп'ютери цього напрямку повинні мати швидкісні зовнішні інтерфейси, володіти високою продуктивністю та надійністю роботи. Часто ці комп'ютери є вбудованими в мобільні засоби, тому додатково вимагається, щоб вони мали малі габарити та низьку споживану потужність.

У курсі «Архітектура комп'ютерів» основну увагу буде приділено вивченню архітектури x86, яка є традиційною та широко використовується в більшості персональних комп'ютерів. Зокрема, ми детально розглянемо особливості побудови та функціонування процесорів компаній Intel та AMD, які тривалий час домінують на ринку настільних і портативних комп'ютерів.

Проте, з урахуванням сучасних тенденцій розвитку комп'ютерної техніки, ми не можемо оминати увагою й архітектуру ARM. Особливо це актуально після того, як компанія Apple у 2020 році почала використовувати власні процесори серії M на базі ARM у своїх комп'ютерах. Це стало вагомим кроком у зміні підходів до побудови високопродуктивних систем, і показало потенціал ARM не лише в мобільних, а й у повноцінних настільних пристроях.

Оскільки Apple наразі є одним із трьох провідних виробників процесорів поряд з Intel та AMD, ми також розглянемо особливості ARM-архітектури, зокрема реалізацію процесорів Apple Silicon. Це дозволить сформувати повніше уявлення про сучасні процесорні архітектури, їхню еволюцію, переваги та застосування у різних типах комп'ютерних систем.

Загалом, функціональні можливості обчислювальних машин обумовлені такими найважливішими техніко-експлуатаційними характеристиками:

- швидкодія, яка вимірюється усередненою кількістю операцій, що виконує обчислювальна машина за одиницю часу;
- розрядність і форми подання чисел, з якими оперує комп'ютер;
- номенклатура, ємність і швидкодія всіх запам'ятовуючих пристроїв;
- номенклатура і техніко-економічні характеристики зовнішніх пристроїв зберігання, обміну і вводу/виводу інформації;
- типи і пропускна здатність пристроїв зв'язку і сполучення вузлів комп'ютера між собою (тип внутрішньомашинного інтерфейсу);
- здатність комп'ютера одночасно працювати з декількома користувачами і виконувати одночасно декілька програм (багатопрограмність);
- типи і техніко-експлуатаційні характеристики операційних систем, що використовуються в обчислювальній машині;
- наявність і функціональні можливості програмного забезпечення;
- здатність виконувати програми, написані для інших типів комп'ютерів (програмна сумісність з іншими типами комп'ютерів);
- система і структура машинних команд;
- можливість підключення до каналів зв'язку і обчислювальної мережі;

- експлуатаційна надійність комп'ютера;
- коефіцієнт корисного використання комп'ютера в часі, визначений співвідношенням часу корисної роботи і часу профілактики.

Обчислювальні машини можуть бути класифіковані за рядом ознак, зокрема: принцип дії; етапи створення та елементна база; призначення; спосіб організації обчислювального процесу; розмір, обчислювальна потужність; функціональні можливості; спроможність до паралельного виконання програм і т.д.

Квантовий комп'ютер

Одним із найбільш революційних досягнень на горизонті є розвиток квантових обчислень. Квантові комп'ютери використовують принципи квантової механіки для виконання обчислень із безпрецедентною швидкістю та масштабом. Хоча, потенціал квантових обчислень все ще перебуває на стадії зародження, потенціал квантових обчислень обіцяє розв'язання складних проблем, які недоступні класичним комп'ютерам. У міру розвитку досліджень ми можемо очікувати на появу нових компонентів, адаптованих до квантових систем, таких як квантові процесори, квантова пам'ять і квантові датчики.

Квантові обчислення – це нова область дослідження, яка зосереджена на розробці комп'ютерних систем, які використовують принципи квантової механіки для виконання обчислень. Традиційні комп'ютери, відомі як класичні комп'ютери, обробляють і зберігають інформацію як двійкові цифри або біти, які представляють або 0, або 1. На відміну від цього, квантові комп'ютери використовують квантові біти, або кубіти, які можуть представляти як 0, так і 1 одночасно завдяки квантовому явищу називається суперпозицією.

Одним із фундаментальних понять у квантових обчисленнях є суперпозиція. У класичних обчисленнях біт може перебувати в тому чи іншому стані, але у квантових обчисленнях кубіт може існувати в суперпозиції станів. Це означає, що кубіт може представляти комбінацію 0 і 1 одночасно, відкриваючи можливість для паралельних обчислень і експоненціально збільшуючи обчислювальну потужність.

Ще одна ключова концепція квантових обчислень – запутаність. Сплутаність дозволяє кубітам стати корельованими таким чином, що стан одного кубіта миттєво пов'язується зі станом іншого, незалежно від відстані між ними. Це явище дозволяє створювати квантові схеми, де обчислення на одному кубіті можуть впливати на стан кількох кубітів, що призводить до складних і потужних операцій.

Квантові обчислення пропонують потенціал розв'язання певних проблем набагато швидше, ніж класичні комп'ютери. Наприклад, він має потенціал зробити революцію в криптографії, зламавши типові алгоритми шифрування, які зараз безпечні через непрактичність розкладання великих чисел. Квантові комп'ютери також можуть посилити проблеми оптимізації,

такі як пошук найбільш ефективних маршрутів або розв'язання складних проблем планування.

Однак, створення практичних квантових комп'ютерів є серйозним науковим та інженерним викликом. Кубіти надзвичайно делікатні та схильні до помилок, спричинених факторами навколишнього середовища або навіть внутрішніми недоліками. Підтримка когерентності, яка є здатністю кубітів залишатися в стані суперпозиції, і мінімізація помилок є напрямками досліджень, які тривають.

Існують різні підходи до створення квантових комп'ютерів, включаючи надпровідні схеми, захоплені іони, топологічні кубіти тощо. Дослідники та компанії в усьому світі, включаючи Google, IBM, Microsoft та інші, активно працюють над удосконаленням технології квантових обчислень та дослідженням її потенційних застосувань.

Хоча квантові обчислення все ще перебувають на ранніх стадіях, вони дають великі надії на розв'язання складних проблем, які обчислювально неможливі для класичних комп'ютерів. Очікується, що в міру розвитку галузі квантові комп'ютери знайдуть застосування в таких сферах, як оптимізація, моделювання квантових систем, відкриття ліків, матеріалознавство тощо.

Оптичні комп'ютери. Фотоний комп'ютер.

Традиційні комп'ютери покладаються на електронні сигнали для обробки та передачі даних, але оптичні обчислення прагнуть використовувати силу світла. Використовуючи фотони замість електронів, оптичні компоненти можуть значно підвищити швидкість передачі даних і покращити енергоефективність.

Дослідники досліджують розробку оптичних компонентів, таких як оптичні процесори, комутатори та з'єднання, щоб подолати обмеження електронних систем. Поява оптичних обчислень має потенціал для революції у високопродуктивних обчисленнях, центрах обробки даних і телекомунікаціях.

Оптичні обчислення – це галузь досліджень і розробок, яка досліджує використання світла й оптики для виконання обчислювальних завдань, а не покладається виключно на традиційні електронні схеми та цифрову обробку. Він спрямований на використання властивостей світла, таких як його висока швидкість, паралельність і потенціал енергоефективності, щоб подолати обмеження електронних обчислювальних систем.

В оптичних обчисленнях інформація кодується та обробляється за допомогою фотонів, які є частинками світла. Замість представлення інформації у вигляді двійкових цифр (бітів), як в електронних обчисленнях, оптичні обчислення можуть використовувати різні властивості світла, такі як інтенсивність, поляризація, довжина хвилі та фаза, для кодування та обробки даних.

Існує кілька підходів і технологій, які досліджуються в оптичних обчисленнях. Вони мають кілька потенційних переваг перед електронними обчисленнями. Світлові сигнали можуть поширюватися з надзвичайно

високою швидкістю, забезпечуючи швидшу обробку та передачу даних. Оптичні системи також можуть виконувати паралельні операції завдяки здатності маніпулювати кількома світловими променями одночасно. Крім того, світлові сигнали відчують менше перешкод і перехресних перешкод, що призводить до покращення цілісності сигналу та потенційно вищої точності обчислень.

Однак в оптичних обчисленнях є проблеми, які необхідно подолати. Управління та маніпулювання світловими сигналами в невеликих масштабах може бути складним, а інтеграція оптичних компонентів з електронними схемами є складним завданням. Оптичні системи також чутливі до таких факторів навколишнього середовища, як зміни температури та вібрації, що може вплинути на надійність обчислень.

Дослідження та розробки оптичних обчислень все ще тривають, і практичні застосування ще не повністю реалізовані. Однак (однак), оптичні обчислення є перспективними для високопродуктивних обчислень, завдань, що містять інтенсивні дані, і спеціалізованих програм, які вимагають масивного паралелізму та високошвидкісної обробки. Це продовжує бути активною сферою дослідження, щоб розсунути межі обчислювальних можливостей за межі традиційних електронних систем.

Біокомп'ютинг

В даний час у пошуках реальної альтернативи напівпровідниковим технологіям створення нових обчислювальних систем учені звертають все більшу увагу на біотехнології, або біокомп'ютинг, який є гібридом інформаційних, молекулярних технологій, також біохімії.

Біокомп'ютинг дозволяє вирішувати складні обчислювальні завдання, користуючись методами, прийнятими в біохімії і молекулярній біології, організовуючи обчислення за допомогою живих тканин, кліток, вірусів і біомолекул. 30 Найбільшого поширення набув підхід, де як основний елемент (процесор) використовуються молекули дезоксирибонуклеїнової кислоти. Центральне місце в цьому підході займає так званий ДНК - процесор. Окрім ДНК як біопроектори можуть бути використані також білкові молекули і біологічні мембрани. Так само, як і будь-який інший процесор, ДНК процесор характеризується структурою і набором команд. У нашому випадку структура процесора - це структура молекули ДНК. А набір команд - це перелік біохімічних операцій з молекулами.

Принцип пристрою комп'ютерної ДНК-пам'яті заснований на послідовному з'єднанні чотирьох нуклеотидов (основного блока ДНК-ланцюга). Три нуклеотиди, з'єднуючись в будь-якій послідовності, утворюють елементарний елемент пам'яті - кодон, які потім формують ланцюг ДНК. Основна трудність в розробці ДНКкомп'ютера пов'язана з проведенням вибірки однокодонних реакцій (взаємодій) усередині ланцюга ДНК. Проте прогрес є вже і в цьому напрямі. Вже є експериментальне устаткування, що дозволяє працювати з одним з 1020 кодонів або молекул ДНК. Іншою проблемою є самотрупвання ДНК, що приводить до втрати інформації. Її

долають введенням в клітину спеціальних інгібіторів - речовин, що запобігають хімічній реакції самогрупування.

Контрольні питання:

1. Коли була заснована компанія IBM і яку роль вона відіграла в історії розвитку комп'ютерної техніки?
2. Які основні досягнення компанії Intel Corporation у галузі процесорів?
3. Чим відома компанія AMD і як вона конкурує з Intel на ринку процесорів?
4. Що спонукало компанію Apple Inc перейти на власні процесори серії M і яку архітектуру вони використовують?
5. Яку роль відіграє ARM Holdings у розвитку сучасних процесорів?
6. У яких пристроях найбільше застосовуються процесори компанії Qualcomm?
7. Чим відрізняються архітектури x86 та ARM з погляду енергоспоживання і продуктивності?
8. Які переваги має ARM-архітектура в мобільних та вбудованих системах?
9. Як розвивалися процесори x86 і які їхні основні риси?
10. Що таке квантовий комп'ютер і як він відрізняється від класичного комп'ютера?
11. Які перспективи має використання квантових комп'ютерів у майбутньому?
12. Що таке оптичний комп'ютер і як він обробляє інформацію?
13. Чим фотонний комп'ютер відрізняється від оптичного і які його переваги?
14. Що розуміють під терміном "біокомп'ютинг" і які матеріали або механізми в ньому використовуються?
15. Які сучасні тенденції розвитку архітектури комп'ютерів спостерігаються у 2020–2025 роках?

ТЕМА 2

ФУНКЦІЇ, СТРУКТУРА ТА ХАРАКТЕРИСТИКИ КОМП'ЮТЕРА

Функції та основні функціональні вузли комп'ютера. Материнська плата. Організація зв'язків між функціональними вузлами комп'ютера. Системна шина. Архітектура шин. Архітектурні моделі IoT пристроїв.

Функції та основні функціональні вузли комп'ютера. Материнська плата.

Розглянемо основу апаратної архітектури комп'ютера, яка з'єднує та координує роботу всіх його основних компонентів. На рис. 2.1 представлено типову схему функціональних вузлів комп'ютера:

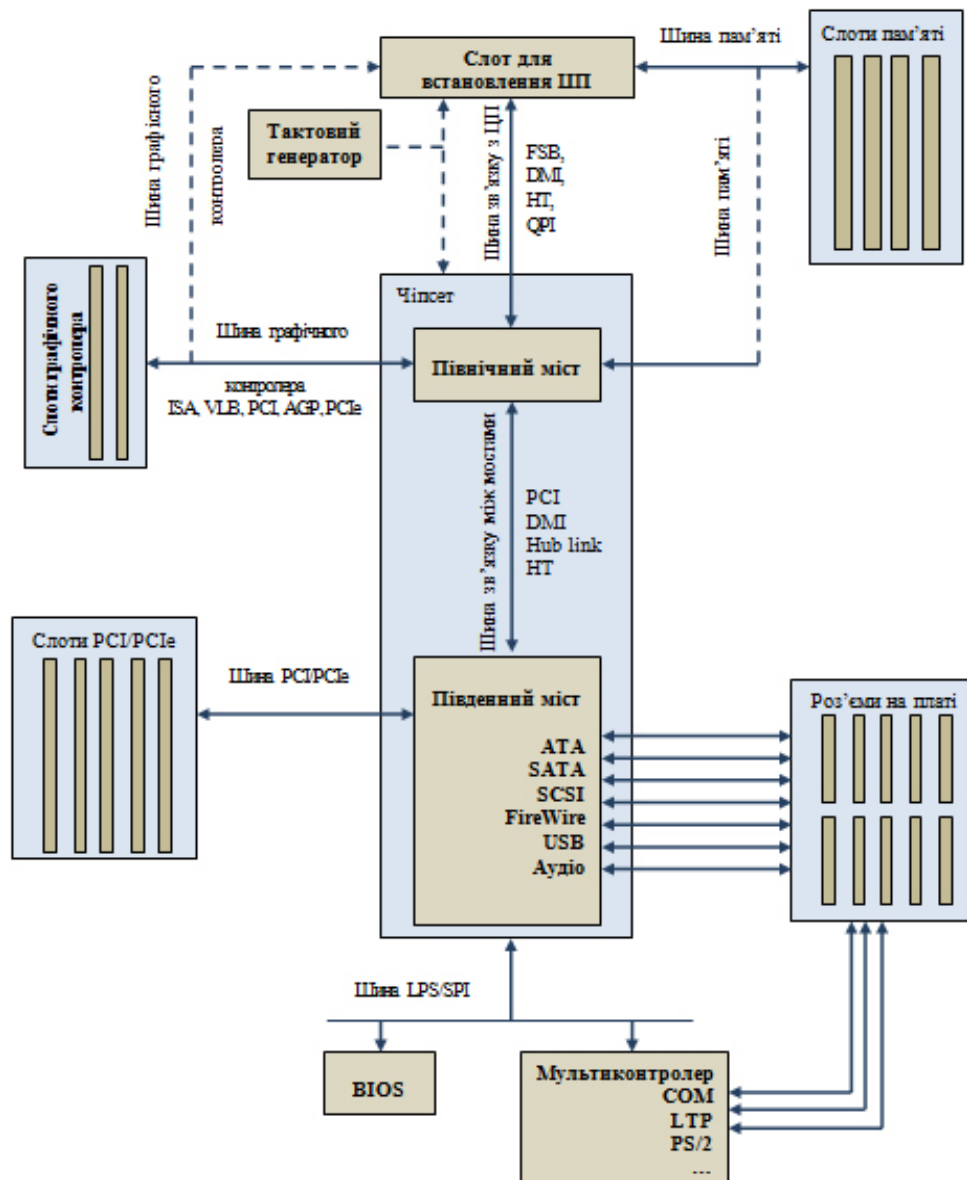


Рисунок 2.1 – Схема функціональних вузлів комп'ютера [3]

Аналізуючи схему, робимо висновок, що комп'ютер складається з п'яти головних, функціонально незалежних частин:

– пристрій введення;

- пристрій виводу;
- пам'ять;
- арифметико-логічний пристрій;
- пристрій керування.

Пристрій введення приймає через цифрові лінії зв'язку закодовану інформацію від операторів, наприклад, через клавіатуру, мишку або від інших комп'ютерів мережі. Отримана інформація або зберігається в пам'яті комп'ютера для подальшого застосування, або негайно використовується арифметичними і логічними схемами для виконання необхідних операцій. Послідовність кроків обробки визначається що зберігається в пам'яті програмою. Отримані результати відправляються назад, у зовнішній світ, за допомогою пристрою виводу. Всі ці дії координуються блоком управління. На рис. 2.1. не показані зв'язки між функціональними пристроями. Пояснюється це тим, що такі зв'язки можуть бути по-різному реалізовані, на цьому ми зупинимося трохи пізніше.

Арифметичні та логічні схеми в комплексі з головними керуючими схемами називають процесором, а все разом узятє обладнання для введення і виведення часто називають пристроєм вводу-виводу (input-output unit).

Тепер звернемося до оброблюваної комп'ютером інформації. Її зручно розділяти на дві основні категорії: команди і дані. Команди, або машинні команди - це явно задані інструкції, які:

- керують пересиланням інформації всередині комп'ютера, а також між комп'ютером і його пристроями вводу-виводу;
- визначають, чи підлягають виконанню арифметичні і логічні операції.

Список команд, що виконують деяку задачу, називається програмою. Зазвичай програми зберігаються в пам'яті. Процесор по черзі витягує команди програми з пам'яті і реалізує обумовлені ними операції. Комп'ютер повністю управляється програмою, що зберігається, якщо не вважати можливість зовнішнього втручання оператора і приєднаних до машини пристроїв введення-виведення. Дані – це числа і закодовані символи, використовувані в якості операндів команд. Проте, термін «дані» часто використовується для позначення будь-якої цифрової інформації. Згідно з цим визначенням, сама програма (тобто список команд) також може вважатися даними, якщо вона обробляється іншою програмою. Прикладом обробки однією програмою інший є компіляція вихідної програми, написаної мовою високого рівня, в список машинних команд, що складають програму на машинній мові, яка називається об'єктною програмою. Вихідна програма надходить на вхід компілятора, який транслює її в програму на машинній мові.

Інформація, призначена для обробки комп'ютером, повинна бути закодована, щоб мати відповідний для комп'ютера формат. Сучасне апаратне забезпечення в більшості своїй засноване на цифрових схемах, у яких є тільки два стійких стани, ON і OFF. В результаті кодування будь-яке число, символ або команда перетворюється в рядок двійкових цифр, званих бітами, кожен з яких має одне з двох можливих значень: 0 або 1.

Схема функціонування вузлів комп'ютера дає загальне уявлення про розташування і способи взаємодії його основних компонент. На практиці усе це реалізовується за допомогою материнської плати.

Материнська плата (системна плата, motherboard) є головною платою ПК, на якій розміщуються всі його основні елементи, лінії з'єднання та роз'єми для підключення зовнішніх пристроїв. Тип встановленої материнської плати визначає загальну продуктивність системи, а також можливості модернізації ПК та підключення додаткових пристроїв [3]. На рис. 2.2 наведено приклад материнської плати, яка розроблена для процесорів Intel Core i9/Core i7/Core i5/Core i3/Pentium Gold/Celeron під сокет LGA1700.

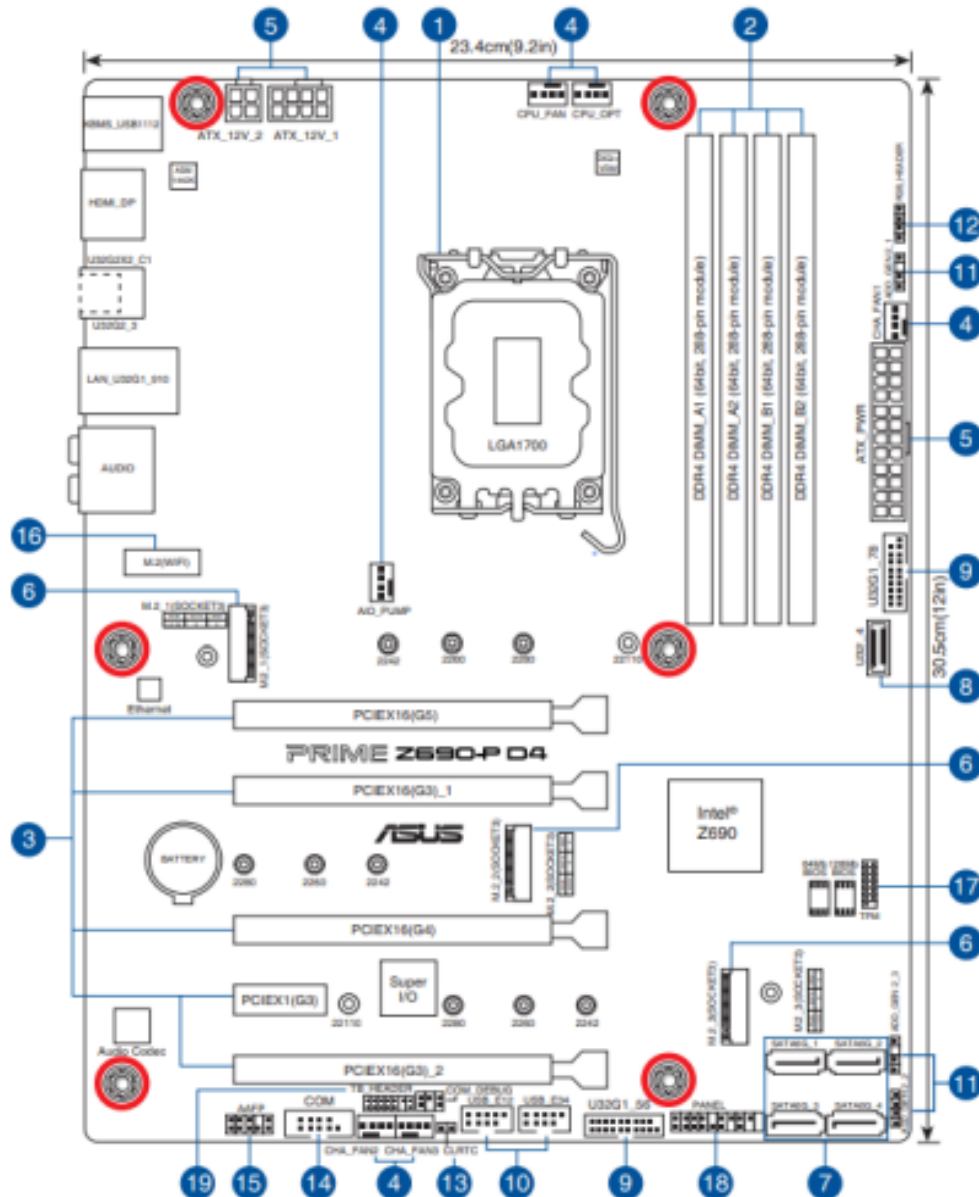


Рисунок 2.2 – Материнська плата PRIME Z690-P D4 компанії Asus [3]

Сокет LGA1700 для процесора (1). Як правило, на процесор встановлюється система охолодження. Роз'єми (слоти) для встановлення модулів оперативної пам'яті (2). Наразі діючим стандартом є 288 контактні модулі DIMM DDR4. Роз'єми (слоти) для встановлення плат розширення (3). На даний час використовуються роз'єми шини PCI Express. Наявність слотів

та можливість встановлення будь-яких карт розширення визначає відкриту архітектуру ПК. Мікросхеми Flash пам'яті (17), в якій зберігаються програми UEFI, POST, програма початкового завантаження операційної системи, початкові налаштування (CMOS Setup). Роз'єми для підключення накопичувачів SATA (11), M2 (6); послідовні порти для підключення периферійних пристроїв USB 3.2 (8-9), USB 2 (10), COM (14). Роз'єми підключення живлення (5), елементів керування передньої панелі корпусу (18), вентиляторів охолодження (4) Всі компоненти материнської плати пов'язані між собою системою провідників, яку називають інформаційною шиною.

Материнські плати характеризуються наступними параметрами.

1. Типи процесорів, які підтримує плата – визначається чипсетом. У наведеному прикладі це процесори Intel Core i9/Core i7/Core i5/Core i3/Pentium Gold/Celeron 12th Gen Intel® Core™ під сокет LGA1700.

2. Типом та кількістю оперативної пам'яті, яку підтримує плата. В нашому випадку це 4 модулі, які працюють в двоканальному режимі, загальним об'ємом до 128 Гігабайт.

3. Кількістю та типом слотів розширення. У наведеному прикладі: 1 x PCIe 4.0 ×16 (x4), 1 × PCI-E 5.0 ×16, 2 × PCI-E 3.0 x16 (x4).

4. Кількістю та типом роз'ємів для підключення накопичувачів. В нашому випадку це 4 × SATAIII та 3 × M2.

5. Форм-фактором. Форм-фактор – це стандарт, що задає габаритні розміри технічного виробу, а також описує додаткові сукупності його технічних параметрів, наприклад форму, типи додаткових елементів, що розміщуються в пристрої, їх положення і орієнтацію.

Системна шина. Архітектура шин.

Шина (магістраль) – це інформаційний канал, який об'єднує всі функціональні блоки і забезпечує обмін даними у вигляді двійкових чисел. Конструктивно шина являє собою n провідників та один спільний провідник (земля). Усі основні блоки МПС з'єднують з єдиною паралельною шиною, яка називається системною шиною SB (**System Bus**). **Системна шина** – це електронний шлях на системній платі, що спільно використовується, до якого підключені всі керовані компоненти ПК.

Системна шина містить три шини: адреси, даних і управління.

Шина адреси ША (Address Bus) – однонапрямлена. Призначена для передавання адреси комірки пам'яті або пристрою введення-виведення. Напрямок передавання по шині адреси – від МП до зовнішніх пристроїв.

Шина даних ШД (Data Bus) - двонапрямлена. Призначена для передавання даних між блоками МПС. Інформація по одних і тих самих лініях ШД може передаватися у двох напрямках – як до МП, так і від нього.

Шина управління ШУ (Control Bus) призначена для передавання керувальних сигналів. Хоча напрям керувальних сигналів може бути різним, однак ШУ не є двонапрявленою, оскільки для сигналів різного напрямку використовуються окремі лінії.

У паралельній шині n бітів передаються по окремих лініях одночасно, у послідовній шині – по єдиній лінії послідовно у часі. Паралельні шини виконують у вигляді плоского кабелю, а послідовні – у вигляді «крученої» пари, коаксіального або волоконно-оптичного кабелю. Коаксіальний кабель використовують для передачі даних на відстань до 100 м, узгоджуючи передавальні та приймальні каскади із хвильовим опором лінії. Волоконно-оптичний кабель використовують для передачі на більші відстані.

Дані по шині можуть передаватися у двох режимах: *синхронному й асинхронному*. У синхронному режимі пристрій керування визначає модулі, що беруть участь в обміні інформацією, синхронізує роботу модулів та керує процесом обміну, виробляючи відповідні сигнали керування і синхронізації. В асинхронному режимі модулі, готові до обміну, ініціюють процес передавання та прийняття інформації, виробляючи відповідні сигнали готовності.

Системна шина характеризується тактовою частотою і розрядністю.

Кількість біт, що одночасно передаються по шині, називається розрядністю шини. Наприклад, якщо шина має розрядність 32 біти, вона може передати 32 біти даних за один такт. Аналогічно, шина з розрядністю 64 біти передає 64 біти за один такт, що збільшує пропускну здатність. Вища розрядність дозволяє передавати більше даних за один тактовий цикл, що покращує продуктивність системи

Тактова частота характеризує число елементарних операцій по передачі даних в 1 секунду. Розрядність шини вимірюється в бітах, тактова частота – у герцах (Hz), зазвичай у мегагерцах (MHz) або гігагерцах (GHz). Це фактично число тактів (імпульсів), що генерує тактовий генератор для синхронізації роботи компонентів.

Вплив тактової частоти і розрядності на продуктивність:

Пропускна здатність (Bandwidth) – це продукт тактової частоти та розрядності, що визначає загальну кількість даних, які можуть бути передані через шину за секунду.

Формула пропускну здатності:

Пропускна здатність = Розрядність × Тактова частота

Наприклад, шина з розрядністю 64 біти і тактовою частотою 1000 МГц (1 ГГц) може передати більше даних за одиницю часу, ніж шина з розрядністю 32 біти і тактовою частотою 500 МГц.

Архітектурні моделі IoT пристроїв

Архітектурні моделі IoT пристроїв (Інтернету речей) визначають, як різні компоненти IoT-системи взаємодіють для збору, передачі, обробки даних і прийняття рішень. Існує кілька загальноприйнятих архітектурних моделей IoT, кожна з яких має свої особливості і відповідає різним вимогам, масштабам і завданням.

Триступенева архітектура (Three-Layer Architecture)

Це одна з найпростіших моделей IoT, яка складається з трьох рівнів:

1. Рівень сприйняття (Perception Layer) – відповідає за збір даних із навколишнього середовища через сенсори та виконавчі механізми (актуатори). Сенсори можуть вимірювати фізичні величини, такі як температура, вологість, світло, тиск, тощо. Пристрої на цьому рівні формують основу IoT і часто мають обмежені обчислювальні можливості.

2. Рівень мережі (Network Layer) – відповідає за передачу зібраних даних з сенсорів до серверів, хмарних систем або інших пристроїв. Для цього використовуються різні протоколи зв'язку: Wi-Fi, Bluetooth, Zigbee, 4G/5G, Ethernet тощо. Також цей рівень забезпечує безпеку передачі даних.

3. Рівень додатків (Application Layer). Включає обробку і використання зібраних даних для конкретних задач (аналітика, автоматизація, прийняття рішень). Наприклад, у "розумному домі" це може бути керування освітленням або клімат-контроль на основі даних з датчиків.

П'ятирівнева архітектура (Five-Layer Architecture)

Ця модель є розширенням триступеневої моделі, додаючи деталі до процесу обробки та управління даними:

1. Рівень сприйняття.

2. Рівень мережі.

3. Рівень обробки (Processing Layer), що відповідає за зберігання та обробку даних в реальному часі або у відкладеному режимі. На цьому рівні зазвичай використовується хмарна обробка або інфраструктура "edge computing".

4. Рівень управління (Management Layer), який забезпечує контроль та управління функціями IoT-системи, наприклад, оновленнями програмного забезпечення, безпекою, конфігурацією пристроїв та моніторингом системи.

5. Рівень додатків.

Архітектура «Edge Computing» (Edge-Centric Architecture)

Ця модель передбачає обробку даних ближче до джерела їхнього виникнення, тобто на самому пристрої або на шлюзі (edge device), замість того, щоб надсилати всі дані до хмари для обробки. Це зменшує затримки і знижує навантаження на мережу.

Архітектура «Fog Computing» (Fog Computing Architecture)

Це розширена модель «Edge Computing», де обробка даних виконується на кількох рівнях між кінцевими пристроями (сенсорами) та хмарою. Це дозволяє забезпечити додаткову обробку даних на рівні мережевих пристроїв (шлюзи, маршрутизатори) перед надсиланням інформації до хмарної інфраструктури.

Архітектура "Хмарних обчислень" (Cloud-Centric Architecture)

У цій архітектурі всі зібрані дані відправляються до хмарних сервісів для зберігання і обробки. IoT-пристрої в такій системі можуть бути відносно простими і не виконувати складних обчислень на місці.

Популярні мікроконтролери/мікропроцесори для IoT-пристроїв, такі як ESP32, ESP8266, STM32, Arduino та Raspberry Pi, використовують різні архітектури залежно від їхніх можливостей і призначення. Їхні архітектури включають як прості однозадачні системи для збирання та передачі даних, так

і складніші обчислювальні платформи для обробки великих обсягів інформації та роботи з нейронними мережами.

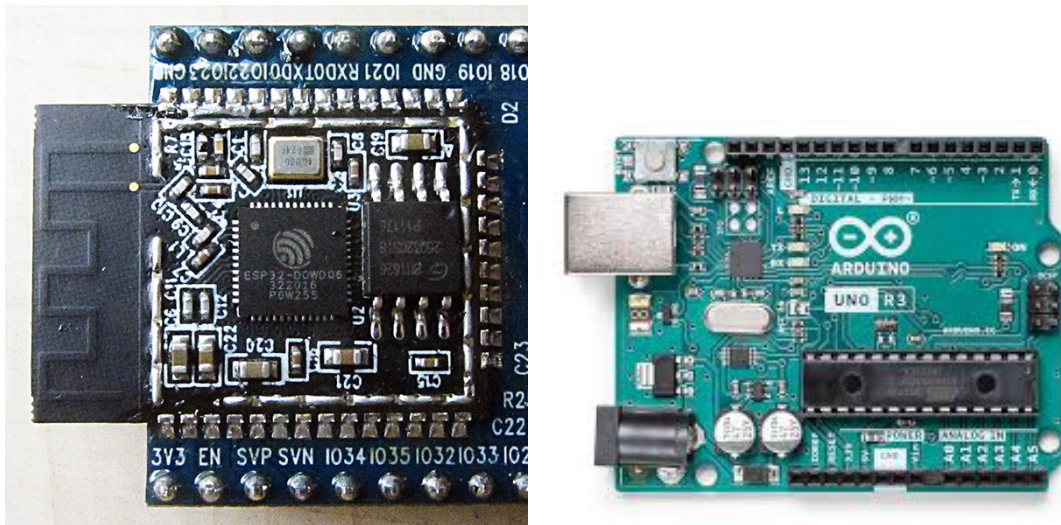


Рисунок 2.3 – Архітектура плат ESP32 та Arduino

Наприклад, архітектура мікроконтролерів для IoT має такі характеристики в розрізі кожного пристрою:

ESP32 має двоядерний процесор і підтримує частоти до 240 МГц, тоді як ESP8266 має однопоточний процесор на частоті до 160 МГц. Вбудовані Wi-Fi та Bluetooth модулі для ESP32 робить їх ідеальними для бездротових IoT-пристроїв. Пам'ять ESP32 має до 512 КБ SRAM і підтримує зовнішні флеш-пам'яті, що дозволяє працювати з великими обсягами даних.

TM32 забезпечує високу продуктивність з частотами до 400 МГц (Cortex-M7). Такі чипи володіють високою енергоефективністю, що робить їх ідеальними для IoT-пристроїв з обмеженим живленням. Великий набір периферійних модулів: SPI, I2C, USB, CAN дозволяє підключати різноманітні пристрої, а підтримка FPU (Floating Point Unit) для обробки з плаваючою точкою дає перевагу цим пристроям для більш складних обчислень.

Arduino Uno з ATmega328 працює на частоті 16 МГц і має 2 КБ SRAM та 32 КБ флеш-пам'яті. Він простий у використанні для невеликих IoT-проектів та підтримує аналогові та цифрові входи/виходи.

Загалом, названі пристрої використовують залежно від можливостей і призначення. Їхні архітектури включають як прості однозадачні системи для збирання та передачі даних, так і складніші обчислювальні платформи для обробки великих обсягів інформації та роботи з нейронними мережами.

Контрольні питання:

1. Які основні функціональні вузли входять до складу комп'ютера і яку роль вони виконують?
2. Що таке материнська плата і які її основні компоненти?
3. Які функції виконує процесор на материнській платі?
4. Яким чином здійснюється організація зв'язків між функціональними вузлами комп'ютера?

5. Що таке системна шина і яку роль вона відіграє у комп'ютерній архітектурі?
6. Які типи шин існують у комп'ютерах і чим вони відрізняються?
7. Назвіть основні характеристики комп'ютерних шин.
8. Які параметри системної шини впливають на швидкість роботи комп'ютера?
9. Що таке архітектура шин і які її основні типи?
10. Як здійснюється передача даних через системну шину?
11. Які основні архітектурні моделі використовуються в IoT пристроях?
12. Які особливості організації архітектури IoT-пристроїв порівняно з традиційними комп'ютерами?
13. Як здійснюється взаємодія між різними компонентами IoT-пристроїв?
14. Що таке шина даних в IoT-системах і як вона взаємодіє з іншими елементами?
15. Як системи IoT використовують шинні архітектури для забезпечення комунікацій між пристроями?

ТЕМА 3.

ТИПИ АРХІТЕКТУР КОМП'ЮТЕРА

Принципи Джона фон Неймана. Принципи двійкового кодування, програмного керування, однорідності пам'яті, адресності. Гарвардська комп'ютерна архітектура. Особливості архітектур мікропроцесорів для збору, обробки та передачі даних. Типи і характеристики комп'ютерних інтерфейсів. Провідні інтерфейси, що застосовуються для взаємодії між різними компонентами всередині IoT-пристроїв.

Принципи Джона фон Неймана

Принципи, згідно з якими функціонує більшість сучасних комп'ютерів, опубліковано в 1946 році американським математиком Джоном фон Нейманом. Він також описав машину, яка може бути універсальним засобом обробки інформації.



Рисунок 3.1 – Джон фон Нейман (1903–1957), основоположник класичної комп'ютерної архітектури

Архітектура фон Неймана (von Neumann architecture) є класичним принципом побудови універсальної обчислювальної машини (комп'ютера), відповідно до якого команди й дані розміщено в одній пам'яті у вигляді двійкового коду, а процесор послідовно вибирає з пам'яті команди й дані, обробляє їх та виконує завдання. Архітектура фон Неймана називається також **Принстонською архітектурою** тому, що її розробка і презентація відбувалася в Принстонському університеті, де працював Джон фон Нейман.

Машина фон Неймана (рис. 3.2), як і практично кожна сучасна ЕОМ загального призначення, складається з чотирьох основних компонентів:

Арифметико-логічний пристрій (АЛП, англ. arithmetic logic unit), який виконує команди з визначеного набору, який називається системою (набором) команд, над порціями інформації, яка зберігається відокремлений від операційного пристрою пам'яті (хоча сучасні архітектури мають в складі операційного пристрою додаткову пам'ять (зазвичай банк регістрів), в якій операнди зберігаються порівняно короткий час безпосередньо в процесі проведення обчислень.

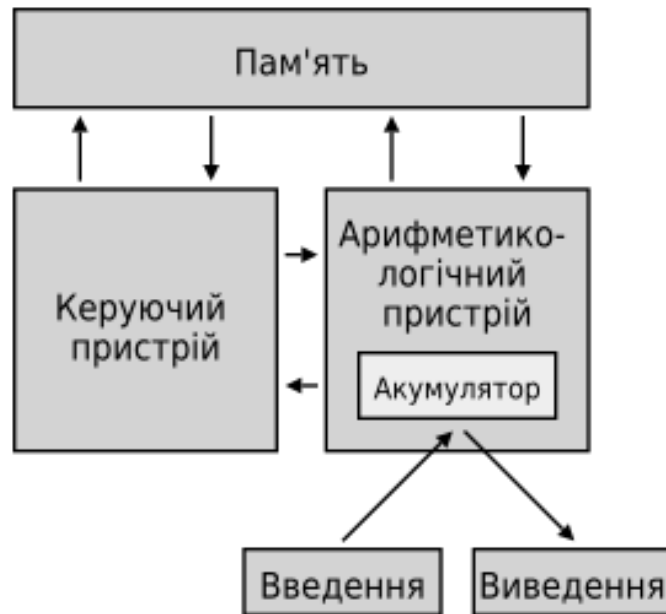


Рисунок 3.2 – Функціональна схема роботи комп'ютера за Джоном фон Нейманом

Пристрій керування процесором (англ. control unit), який організує послідовне виконання алгоритмів, розшифрування команд, які надходять із пристрою пам'яті (див. нижче), реагує на аварійні ситуації та виконує загальні функції керування всіма вузлами обчислювальної машини. Зазвичай АЛП та пристрій керування об'єднуються в структуру, яка називається центральним процесором. Слід звернути увагу, що вимога саме послідовного, в порядку надходження з пам'яті (в порядку зміни адрес в лічильнику команд) виконання команд є принциповою. Архітектури, які не дотримуються такого принципу, взагалі не вважаються фон-нейманівськими.

Комп'ютерна пам'ять (англ. primary storage) – масив комірок з унікальними ідентифікаторами (адресами), в яких зберігаються команди та дані.

Пристрої вводу/виводу, які забезпечують зв'язок ЕОМ з зовнішнім світом, передають інформацію на переробку в ЕОМ та приймають результати.

Принципи двійкового кодування, програмного керування, однорідності пам'яті, адресності.

1. **Принцип двійкового кодування.** Вся інформація, що обробляється у комп'ютері, представлена у двійковому коді, що добре відображає два стійких стани елементів комп'ютера (є імпульс – «0», немає імпульсу – «1»)

2. **Принцип програмного керування.** Робота на комп'ютері здійснюється за допомогою програм. Програма складається з набору команд, які в певній послідовності виконуються процесором. Ефективність програмного керування є вищою, коли програму можна застосовувати багаторазово і за різних початкових даних.

3. Принцип однорідності пам'яті. Програми і дані зберігаються в одній пам'яті, тому комп'ютер не розрізняє, що знаходиться у комірці пам'яті: число, текст або команда. Над командами можна виконувати ті ж самі дії, що й над даними. Наприклад, команди одної програми можуть бути отримані як результат виконання іншої програми.

4. Принцип адресності. Пам'ять комп'ютера складається з пронумерованих комірок. Процесор в будь який момент може звернутися до будь-якої комірки за заданою адресою.

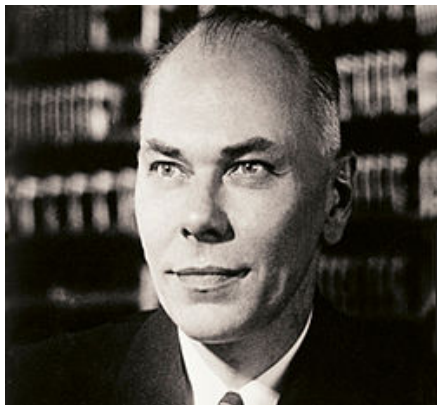
Звісно, дана архітектура підлягала неабиякій критиці, в основному. Хоча це не стосувалося безпосередньо принципів фон Неймана, але часто апелювали саме до «класичної архітектури фон Неймана» в критиці її досить примітивного та низькорівневого набору команд, який, на думку критиків, абсолютно не відповідає сучасному стану справ в індустрії розробки програмного забезпечення, зокрема в наявності мов високого рівня, які набагато підвищують продуктивність праці програміста за рахунок пропонування йому більш високорівневих абстракцій, і потрібно зазвичай до декількох сот машинних команд замість однієї команди мови високого рівня.

Розділення операційного пристрою та пам'яті в класичній архітектурі фон-Неймана вважається її суттєвим недоліком. Полюбляють казати про так звану «шийку пляшки» фон-нейманівської архітектури (термін, запропонований Джоном Бекусом (John Backus) в 1977. Ця «шийка» виникає між операційним пристроєм (центральною процесором) і пам'яттю, адже швидкість обробки інформації в процесорі зазвичай є набагато більшою, ніж швидкість роботи пристрою пам'яті, який не встигає забезпечувати процесор новими порціями інформації, що призводить до простоїв. Проблема вирішується за рахунок побудови складнішої ієрархії пам'яті, зокрема введенням кеш-пам'яті (швидшої, але й дорожчої за основну). Іншим прикладом часткового вирішення цієї проблеми є **гарвардська архітектура**, в якій пам'ять команд та даних розділена, що дозволяє інтенсифікувати обмін між пристроєм пам'яті та центральним процесором.

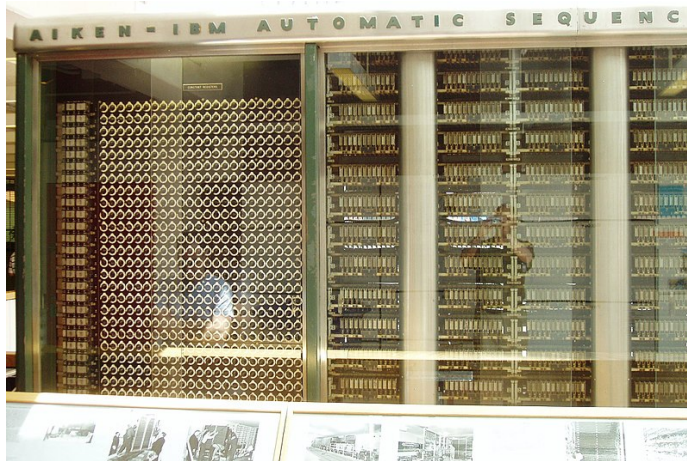
Гарвардська комп'ютерна архітектура

У 1930-х роках уряд США доручили Гарвардському та Принстонському університету Розробити архітектуру ЕОМ для військово-морської артилерії. Наприкінці 1930-х років у Гарвардському університеті Говардом Ейкеном (рис. 3.4) була розроблена архітектура комп'ютера-1, надалі звана по імені цього університету [4].

У жовтні 1937 року оригінальна ідея була продемонстрована Ейкеном для компанії IBM. Однак перемогла більш проста у реалізації розробка Принстонського університету, відома як архітектура фон Неймана.



а)



б)

Рисунок 3.4 – а) Г.Ейкен (Howard Hathaway Aiken, 1900-1973), засновник Гарвардської архітектури; б) фрагмент комп'ютера Марк-1 [4]

Класична гарвардська архітектура полягала у наступному (рис. 3.5). Типові операції (додавання та множення) вимагають від будь-якого обчислювального пристрою кількох дій:

- вибірку двох операндів;
- вибір інструкції та її виконання;
- збереження результатів.

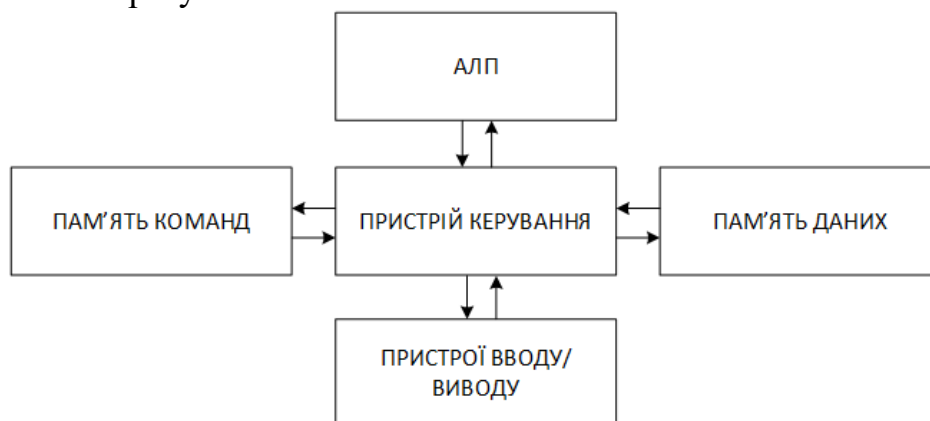


Рисунок 3.5 – Функціональна схема роботи комп'ютера за Говардом Ейкеном

Ідея, реалізована Ейкеном, полягала у фізичному поділі ліній передачі команд та даних. У першому комп'ютері Ейкена «Марк-1» для зберігання інструкцій використовувалася перфострічка, а для роботи з даними – електромеханічні регістри. Це дозволяло одночасно пересилати й обробляти команди і дані, завдяки чому значно підвищувалася загальна швидкодія комп'ютера.

У гарвардській архітектурі характеристики пристроїв пам'яті для інструкцій і пам'яті для даних не обов'язково повинні бути однаковими. Зокрема, ширина слова, тактування, технологія реалізації та структура адрес пам'яті можуть розрізнятися. У деяких системах інструкції можуть зберігатися в пам'яті тільки для читання, в той час як для збереження даних зазвичай потрібно пам'ять з можливістю читання і запису. У деяких системах потрібно

значно більше пам'яті для інструкцій, ніж пам'яті для даних, оскільки дані зазвичай можуть довантажуватися з зовнішньої або більш повільної пам'яті. Така потреба збільшує бітність (ширину) шини адреси пам'яті інструкцій в порівнянні з шиною адреси пам'яті даних.

Відмінність від архітектури фон Неймана

У чистій архітектурі фон Неймана процесор в кожен момент часу може або читати інструкцію, або читати/записувати одиницю даних з/в пам'ять. Обидві дії одночасно відбуватися не можуть, оскільки інструкції і дані використовують один і той же потік (шину). У комп'ютері з використанням гарвардської архітектури процесор може читати інструкції і виконувати доступ до пам'яті даних одночасно, без використання кеш-пам'яті. Таким чином, комп'ютер з гарвардською архітектурою при певній складності схеми працює швидше, ніж комп'ютер з архітектурою фон Неймана, оскільки шини інструкцій і даних розташовані на різних, не пов'язаних між собою фізично, каналах.

Окрім Принстонської (фон Нейманової) та Гарвардської архітектур, існують ще кілька альтернативних або спеціалізованих архітектур, які використовуються для різних обчислювальних систем. Це, наприклад, архітектури Хокні (Hokey) і Джесхоупа (Jeshope), які є менш відомими, але також цікавими концепціями в обчислювальних системах. Архітектура Хокні спрямована на оптимізацію роботи паралельних обчислювальних систем, а архітектура Джесхоупа була розроблена для підвищення гнучкості та адаптивності обчислювальних систем, зокрема в умовах обробки даних з високою швидкістю.

Особливості архітектур мікропроцесорів для збору, обробки та передачі даних.

Загальна структура мікропроцесорної системи подана на рисунку 3.6. Принципи організації такої системи передбачають:

1) функціональну закінченість модулів та блоків системи, які намагаються, щоб виконання зовнішньої або внутрішньої функції забезпечувалося тільки переважно одним блоком або модулем. Особливість – простота налагоджування, модернізації й розширення.

2) регулярність структури. Це означає, що для побудови системи переважно застосовують інформаційні системи регулярної структури (пам'ять, контролери інтерфейсів, системні контролери тощо), а у випадку застосованих нестандартних модулів – програмовані логічні інтегральні схеми ПЛІС. Особливість – простота проектування й налагоджування, швидкість розробки.

3) шинну організацію зв'язку. Блоки, модулі системи з'єднуються між собою за допомогою шин даних, адреси й управління, що призводить до необхідності застосування адрес (імен) пристроїв (так званий адресний принцип). Особливість – зниження апаратних витрат й масогабаритних показників, простота розширення.

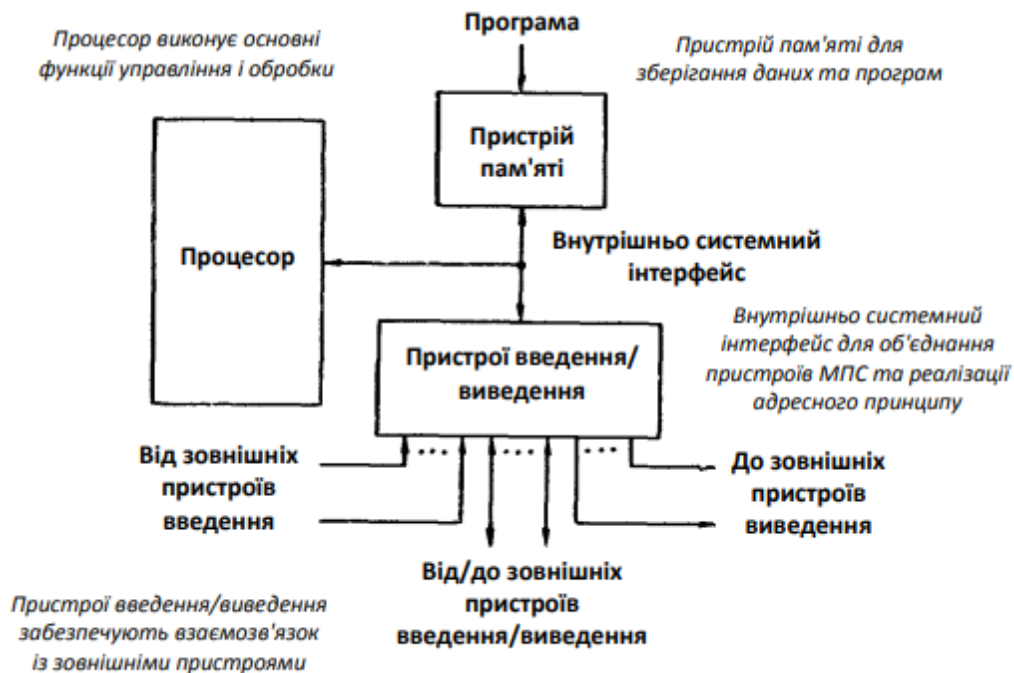


Рисунок 3.6 – Загальна структура мікропроцесорної системи [5]

Розглянемо кілька прикладів архітектур мікропроцесорів для збору, обробки та передачі даних [5]:

Приклад 1. Система автоматичного збору метеорологічної інформації

Завданням системи є доставка інформації від датчиків в центр збору даних. На рівні цільового завдання визначається:

- максимальна кількість датчиків;
- тип і кількість інформації для кожного датчика;
- терміни збору інформації (наприклад: безперервний збір, період збору та ін.) для кожного датчика і типу інформації;
- спосіб передачі інформації для кожного датчика - по ініціативи датчика або центру;
- додаткові функції системи, що не належать безпосереднім але до вирішення основного завдання збору метеорологічної інформації;
- інші характеристики, важливі для функціонування системи в цілому.

Приклад 2. Система оповіщення населення про надзвичайні ситуації

Завданням системи є доведення сигналів оповіщення до об'єктів оповіщення.

На рівні цільового завдання визначається:

- максимальна кількість сигналів оповіщення;
- максимальна кількість об'єктів оповіщення;
- спосіб доведення сигналів – звуковий (сирена, мова і т.п.), візуальний («мигалка», дисплей, біжучий рядок і т.ін.);
- терміни доведення сигналу від джерела до кожного об'єкта;
- способи підтвердження об'єктом отримання сигналу оповіщення;
- додаткові функції системи, що не відносяться безпосередньо до вирішення основного завдання оповіщення;

- інші характеристики, важливі для функціонування системи в цілому.

У цих двох прикладах – системи автоматичного збору метеорологічної інформації та системи оповіщення населення про надзвичайні ситуації – розглядаються різні архітектурні підходи до збору, обробки та передачі даних. Хоча їхні цільові завдання різні, обидві системи демонструють ключові принципи, які використовуються в мікропроцесорних архітектурах для цих цілей. Обидві системи демонструють, як мікропроцесорні архітектури для збору, обробки та передачі даних можуть бути адаптовані для конкретних цілей. Важливими є гнучкість архітектури, здатність обробляти та передавати дані ефективно, з урахуванням різних джерел і способів передачі інформації.

Типи і характеристики комп'ютерних інтерфейсів.

Тлумачний словник обчислювальних систем визначає поняття інтерфейс (interface) як кордон розділу двох систем, пристроїв або програм; елементи з'єднання та допоміжні схеми управління, що використовуються для з'єднання пристроїв.

В світі комп'ютерної техніки інтерфейс – це фізичні пристрої, що забезпечують зв'язок між двома іншими пристроями, що дозволяють підключати до персональних (і не тільки) комп'ютерів різноманітні периферійні пристрої і їх контролери.

Відповідно до функціонального призначення інтерфейси можна поділити на такі основні класи:

- системні інтерфейси комп'ютера;
- інтерфейси периферії (загальні й спеціалізовані);
- інтерфейси програмно-керованих модульних систем і приладів;
- інтерфейси мереж передачі даних і ін.

Системні інтерфейси комп'ютера – це *контролери, слоти, шини та порти*.

Розглянемо конкретні приклади сучасних інтерфейсів. [6]

PCI Express (PCIe, PCI-e) – це один з найпоширеніших протоколів передачі даних. Він використовується в сучасній комп'ютерній техніці для забезпечення взаємодії різних її функціональних блоків між собою. PCI Express (Peripheral Component Interconnect Express, скорочено - PCIe або PCI-e) – це комп'ютерна шина, яка використовує високопродуктивний протокол послідовної передачі даних. Передача даних здійснюється через один комунікаційний канал. Висока швидкість досягається за рахунок великої частоти передачі даних. А для пристроїв, які потребують особливо високих швидкостей обміну даними, одночасно використовується кілька таких каналів (ліній). Наприклад, сучасні ігрові відеокарти підключаються до комп'ютера через 16 ліній PCIe (PCIe x16).

У версіях PCIe 1.0 і PCIe 2.0 використовується схема кодування 8b/10b. Тобто, кожен 8-бітний блок кодується в 10-бітний, в якому тільки 80% переданих даних є корисними. Решта 20% потрібні для забезпечення належного функціонування протоколу. У PCIe 3.0 і новіших її версіях дані кодуються за більш ефективною схемою 128b/130b (кожні 128 біт кодуються

в 130-бітний блок). Частка корисного вмісту в даних, що передаються, тут становить вже близько 98,46%.

Різні версії PCIe відрізняються не тільки способом «ущільнення» бітів в блоки, а й частотою передачі даних. У PCIe 1.0 вона складає 2,5 ГТ/с (гігатранзакцій в секунду), тобто за одну секунду передається 2,5 мільярди бітів. Для кращого сприйняття переведемо це в звичні одиниці:

$$2,5 \cdot 10^9 \text{ Біт / с} = 312,5 \text{ Мегабайт/с}$$

Враховуючи, що лише 80% з них є корисними даними, реальна пропускна здатність PCIe 1.0 складає 250 Мегабайт / с.

У PCIe 5.0 частота передачі даних зросла аж до 32 ГТ / с. Переведемо це в зручний вид:

$$32 \cdot 10^9 \text{ Біт / с} = 4000 \text{ Мегабайт / с} = 4 \text{ Гігабайт / с}$$

Оскільки корисні дані становлять 98,46%, реальна пропускна здатність PCIe 5.0 дорівнює 3,938 Гігабайт / с.

Детальніше особливості різних версій PCIe подано у таблиці 3.1:

Таблиця 3.1 – Характеристики шин PCIe [6]

Версія PCI Express	Рік виходу	Схема кодування	Швидкість передачі	Пропускна здатність на x ліній:			
				x1	x4	x8	x16
PCIe 1.0	2002	8b/10b	2,5 ГТ/с	250 МБ/с	1 ГБ/с	2 ГБ/с	4 ГБ/с
PCIe 2.0	2007	8b/10b	5 ГТ/с	500 МБ/с	2 ГБ/с	4 ГБ/с	8 ГБ/с
PCIe 3.0	2010	128b/130b	8 ГТ/с	984,6 МБ/с	3,94 ГБ/с	7,88 ГБ/с	15,8 ГБ/с
PCIe 4.0	2017	128b/130b	16 ГТ/с	1,969 ГБ/с	7,88 ГБ/с	15,8 ГБ/с	31,5 ГБ/с
PCIe 5.0	2019	128b/130b	32 ГТ/с	3,938 ГБ/с	15,75 ГБ/с	31,5 ГБ/с	63 ГБ/с

Версії і кількість ліній PCIe в різних моделях процесорів і чіпсетів відрізняється. Значна їх частина формується в роз'єми, які розміщуються на материнській платі. Вони дають можливість підключати до комп'ютера різноманітні пристрої (відеокарти, звукові карти, мережеві карти, Wi-Fi-адаптери та ін.).

На материнській платі сучасного комп'ютера можна знайти роз'єми PCIe декількох видів, які відрізняються кількістю використовуваних в них ліній PCIe (від x1 до x16 ліній).

Незалежно від того, наскільки старим є комп'ютер, і яка версія PCIe в ньому використовується, ці роз'єми завжди виглядають однаково (рис. 3.7).

Лінії PCIe використовуються також для створення деяких інших роз'ємів, а саме, роз'ємів M.2 (служать для підключення сучасних запам'ятовуючих пристроїв, а також пристроїв деяких інших типів). Слоти і карти формату M.2 (раніше цей формат називався Next Generation Form Factor - NGFF) спочатку розроблялися як більш швидкісна і компактніша заміна для mSATA – популярного стандарту, використовуваного твердотільних накопичувачами в різних мобільних платформах.

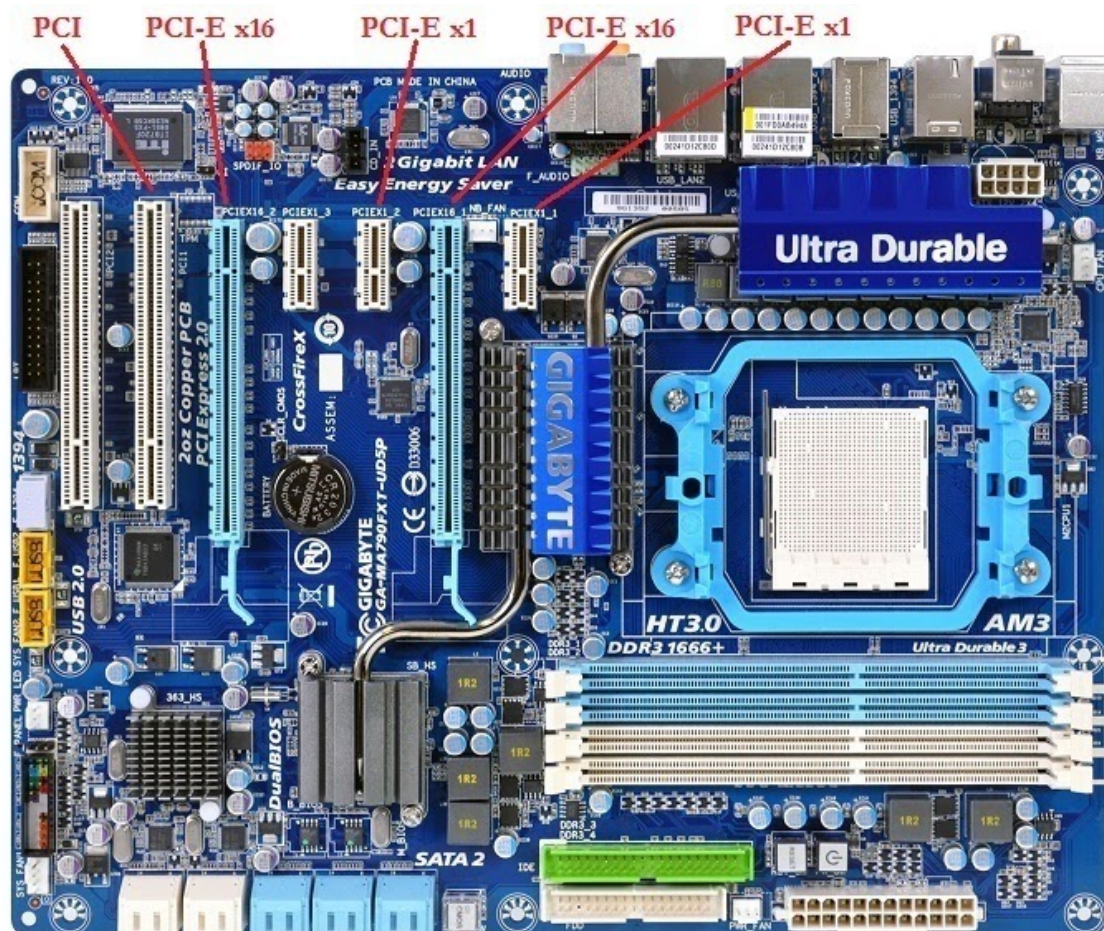


Рисунок 3.7 – Типи PCIe на системній платі комп'ютера [6]

На відміну від свого попередника, M.2 (рис. 3.8) пропонує принципово більшу гнучкість як у логічній, так і в механічній частині.

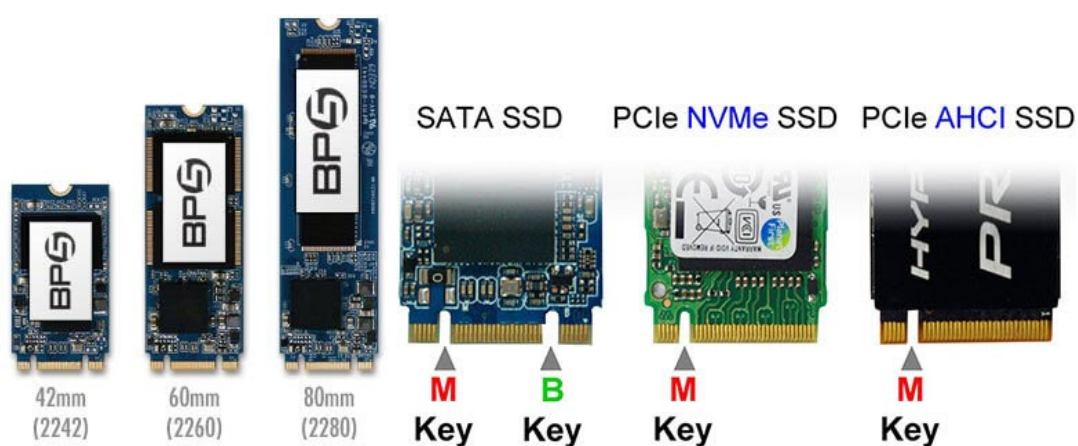


Рисунок 3. 8 – Роз'єми M2 [6]

Новий стандарт описує кілька варіантів довжини та ширини карт, а також дозволяє використовувати для підключення твердотільних накопичувачів як SATA, так і швидкісний інтерфейс PCI Express. Усі материнські плати, оснащені слотом M.2, використовують шину PCI-E x4.

SATA (Serial ATA) – послідовний інтерфейс обміну даними з накопичувачами інформації (як правило, з жорсткими дисками). SATA є

подальшим розвитком інтерфейсу ATA (IDE), який після появи SATA був перейменований в PATA (Parallel ATA).

SATA/150. На початку стандарт SATA передбачав роботу шини на частоті 1,5 ГГц, що забезпечує пропускну здатність приблизно в 1,2 Гбіт/с (150 МБ/с). (20%-а втрата продуктивності пояснюється використанням системи кодування 8B/10B, при якій на кожні 8 біт корисної інформації припадає 2 службових біта). Пропускна здатність SATA/150 незначно вища пропускну здатності шини Ultra ATA (UDMA/133). Головною перевагою SATA перед PATA є використання послідовної шини замість паралельної. Незважаючи на те, що послідовний спосіб обміну принципово повільніший паралельного, у цьому випадку це компенсується можливістю роботи на більш високих частотах за рахунок більшої перешкодостійкості кабелю. Це досягається:

- меншим числом провідників;
- об'єднанням інформаційних провідників в 2-і виті пари, екрановані заземленими провідниками.

SATA/300. Стандарт SATA/300 працює на частоті 3 ГГц, забезпечує пропускну здатність до 2,4 Гбіт/с (300 МБ/с). Уперше був реалізований у контролері чіпсету nForce 4 фірми NVIDIA. Досить часто стандарт SATA/300 називають SATA II або SATA 3.0. Теоретично SATA/150 й SATA/300 пристрої повинні бути сумісні (як SATA/300 контролер і SATA/150 пристрій, так і SATA/150 контролер й SATA/300 пристрій) за рахунок підтримки узгодження швидкостей (у меншу сторону), однак для деяких пристроїв і контролерів потрібне ручне виставлення режиму роботи (наприклад, на жорсткому диску фірми Seagate, що підтримують SATA/300, для примусового включення режиму SATA/150 передбачений спеціальний джампер).

Стандарт SATA передбачає можливість збільшення швидкості роботи до 600 МБ/с (6 ГГц). SATA використовує 7-контактний роз'єм замість 40-контактного роз'єму в PATA. SATA-кабель має меншу ширину, за рахунок чого зменшується опір повітрю, що обдуває комплектуючі комп'ютера; поліпшується охолодження системи.

SATA-кабель за рахунок своєї форми більш стійкий до багаторазового підключення. Шнур живлення SATA так само розроблений з урахуванням багаторазових підключень. Роз'єм живлення SATA подає 3 напруги: +12 В, +5 В і +3,3 В; однак сучасні пристрої можуть працювати без напруги +3,3 В, що дає можливість використати пасивний перехідник зі стандартного роз'єму IDE на SATA. Ряд SATA пристроїв поставляється із двома роз'ємами живлення: SATA й Molex.

Стандарт SATA відмовився від традиційного для PATA підключення по два пристрої на шлейфу; кожному пристрою покладається окремий кабель, що знімає проблему неможливості одночасної роботи пристроїв, що перебувають на одному кабелі (і затримок, що виникали звідси), зменшує можливі проблеми при зборці (проблема конфлікту Slave/Master пристроїв для SATA відсутній), усуває можливість помилок при використанні нетермінованих PATA-шлейфів.

eSATA – це інтерфейс підключення зовнішніх пристроїв, що підтримує режим "гарячої заміни" (англ. Hot-plug). Був створений трохи пізніше SATA (у середині 2004).

У eSATA середня практична швидкість передачі даних вища, ніж у USB 2.0 або IEEE 1394. Сигнально SATA та eSATA сумісні. Однак їм потрібні різні рівні сигналу. Також для підключення необхідні два кабелі: шина даних та кабель живлення. Планується відмовитись від окремого кабелю живлення для виносних eSATA-пристроїв. Роз'єми в нього більш надійні та екрановані і конструктивно розраховані на більше підключень, ніж SATA. Однак фізично вони несумісні зі звичайними SATA (рис. 3.9).

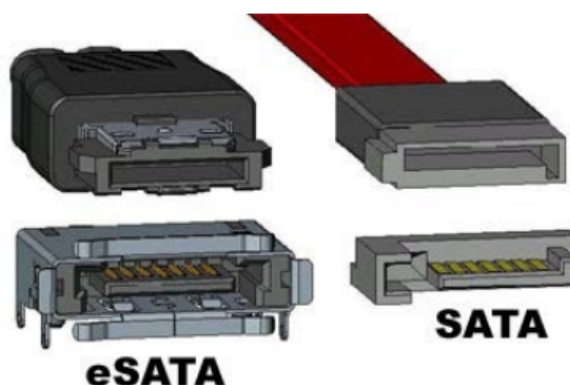


Рисунок 3. 9 – Роз'єми SATA [6]

Високошвидкісний інтерфейс USB 3.0

Якщо говорити про основну перевагу стандарту USB 3.0 порівняно зі стандартом USB 2.0, то це, звичайно ж, швидкість передачі даних. Специфікацією USB 3.0 передбачений режим SuperSpeed зі швидкістю передачі даних до 5 Гбіт/с (640 Мбайт/с), тобто більш ніж у 10 разів, що перевищує ту, що передбачена специфікацією USB 2.0. Звичайно, реальна швидкість передачі даних, швидше за все, буде нижчою, проте навіть з урахуванням цього для більшості периферійних пристроїв такої пропускної здатності шини USB 3.0 більш ніж достатньо. Висока швидкість передачі у шини USB 3.0 – це не єдина її відмінність від шини USB 2.0. При всіх відмінностях (яких багато) зберігається зворотна сумісність USB 3.0 з USB 2.0. Тобто в специфікації USB 3.0, окрім режиму SuperSpeed, залишені і режими Hi-Speed, Full-Speed та Low-Speed. Крім того, USB 3.0 сумісний з USB 2.0. Це означає, що всі периферійні пристрої з USB 2.0 можна буде підключати до шини USB 3.0 [6].

USB 3.1. На той час, як з'явилася версія 3.0, USB став загальновизнаним стандартом. У 2013 році з'явився стандарт USB 3.1 відомий як SuperSpeed+, який подвоїв швидкість до 10 Гбіт/с при використанні роз'ємів USB типу A та USB-C [6].

USB 3.2. Випущений у вересні 2017 року, USB 3.2 дозволяє сумісним пристроям використовувати швидкість передачі даних SuperSpeed (5 Гбіт/с) або SuperSpeed + (10 Гбіт/с). Однак у цій останній версії також з'явився новий

режим SuperSpeed + (20 Гбіт/с), який активується при використанні роз'єму USB-C та останніх моделей пристроїв (рис. 3.10).



Рисунок 3.10 – Різновиди USB [6]

Thunderbolt (у перекладі з англійської – удар блискавки) – це новий високошвидкісний інтерфейс для підключення пристроїв до комп'ютера, раніше відомий як Light Peak. Цей інтерфейс розробили компанія Intel, технічним партнером якої стала компанія Apple, яка оснащує свої ноутбуки MacBook Pro роз'ємом Thunderbolt. Intel позиціонує цей інтерфейс як заміну у майбутньому нинішніх дротових інтерфейсів USB, SCSI, SATA та FireWire. Спочатку для передачі даних планувалося використовувати тільки оптоволокно, але це виявилось недешевим задоволенням і тому в першому варіанті роз'єму використовується мідь, яка, як виявилось, теж добре працює. У зв'язку з цим, робоча назва Light Peak стала неактуальною і її замінили на Thunderbolt. Проте Intel і зараз продовжує працювати над оптоволоконним варіантом даного інтерфейсу, який, ймовірно, з'явиться в недалекому майбутньому. Головні переваги оптики - велика пропускна спроможність та велика відстань передачі сигналу. А мідний кабель Thunderbolt не може бути завдовжки понад 3 метри. Thunderbolt може підключати периферійні пристрої, що споживають трохи більше 10 Вт потужності. Найголовнішою перевагою Thunderbolt є можливість підключати пристрої послідовно, внаслідок чого можна буде позбутися звичної купи проводів на столі. Пропускна здатність шини Thunderbolt на даний момент складає 10 Гбіт/с в обох напрямках. Це вдвічі перевищує швидкість передачі даних порту USB 3.0. Однак Intel не позиціонує Thunderbolt як конкурента USB і обіцяє, що чіпсети нового покоління будуть йти з підтримкою USB 3.0 [6].

У 2022 році відбувся важливий крок у розвитку архітектури материнських плат, зокрема з появою оперативної пам'яті стандарту DDR5 та інтерфейсу PCI Express 5.0. Ці технології забезпечили значне збільшення пропускної здатності та ефективності обміну даними, що особливо важливо

для сучасних високопродуктивних систем. DDR5, порівняно з DDR4, пропонує вищу тактову частоту, більшу щільність модулів пам'яті та покращене енергоспоживання.

PCIe 5.0, у свою чергу, подвоїв пропускну здатність у порівнянні з попередньою версією PCIe 4.0, що дозволяє підключати високошвидкісні пристрої, зокрема графічні карти, SSD-диски нового покоління та спеціалізовані плати розширення. Поява цих інтерфейсів стала можливою завдяки оновленню апаратної платформи процесорів Intel Alder Lake та AMD Ryzen 7000, що підтримують відповідні контролери. Зокрема, процесори Intel Arrow Lake-S 2023 року отримали двоканальний контролер оперативної пам'яті DDR5 з підтримкою конфігурації 2DPC (два модулі на канал), 16 ліній PCI Express 5.0 для дискретного відеоадаптера, чотири PCIe 5.0 для високошвидкісного NVMe-накопичувача та ще чотири лінії PCIe 4.0 для SSD або інших пристроїв. Також варто відзначити інтегровану підтримку інтерфейсу Thunderbolt 4 й можливість виведення зображення з вбудованої графіки через порти HDMI 2.1 і DisplayPort 2.1. (рис. 3.11)



Рисунок 3.11 – PCIe 5.0 та DDR5 на архітектурі Intel Alder Lake [7]

Провідні інтерфейси, що застосовуються для взаємодії між різними компонентами всередині IoT-пристроїв

У IoT-пристроях для взаємодії між різними компонентами використовуються специфічні інтерфейси, які забезпечують обмін даними між мікроконтролерами, сенсорами, модулями зв'язку та іншими периферійними пристроями. Основні вимоги до інтерфейсів IoT-пристроїв включають енергоефективність, компактність, низькі витрати на апаратне забезпечення і можливість роботи в умовах обмежених ресурсів. Розглянемо типові периферійні інтерфейси.

SPI (Serial Peripheral Interface)

Це – швидкісний інтерфейс для взаємодії між мікроконтролерами і периферійними пристроями, такими як дисплеї, пам'ять, сенсори, мікросхеми

зв'язку. Інтерфейс використовується для роботи з різними периферійними пристроями, наприклад, це можуть бути різні ЦАП / АЦП, цифрові потенціометри, різні датчики, розширювачі портів введення / виводу (GPIO), Flash і EEPROM пам'ять і навіть більш складна периферія, така як, наприклад, звукові кодеки і контролери Ethernet і ін. (рис. 3.12).

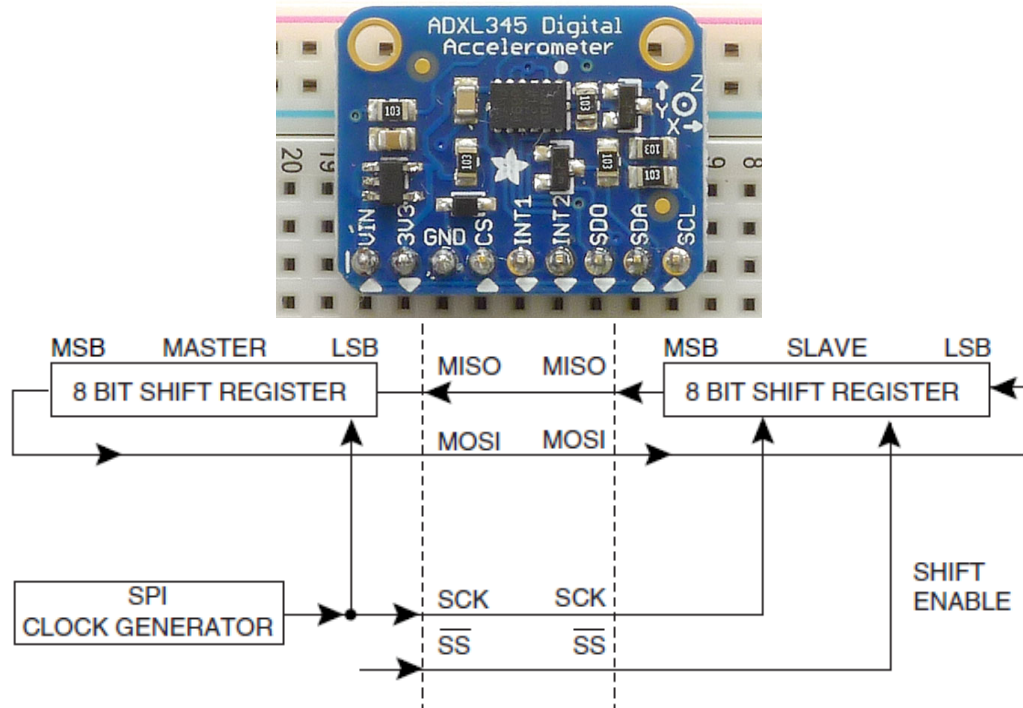


Рисунок 3.12 – Інтерфейс SPI: зовнішній вигляд та структура шини [6]

Інтерфейс SPI має чотири сигнальні лінії:

Лінія MOSI (Master-Out, Slave-In) - вихідна лінія даних ведучого інтерфейсу і входна лінія даних веденого інтерфейсу. З назви виходить, що лінія призначена для передачі даних від ведучого (Master) інтерфейсу (або вузла мережі) до веденого (Slave) інтерфейсу (або вузлу мережі).

Лінія MISO (Master-In, Slave-Out) - входна лінія даних ведучого інтерфейсу і вихідна лінія даних веденого інтерфейсу. Лінія призначена для передачі даних від веденого інтерфейсу до ведучого. Дані передаються байтами, побітно, починаючи із старшого біта. Слід пам'ятати, що вивід MISO веденого інтерфейсу знаходиться в стані високого імпедансу, якщо ведений інтерфейс не вибраний по лінії NSS

Лінія NSS (Slave Select) – (SS) лінія вибору веденого.

Лінія SCK (Serial Clock) - вихідна лінія тактових імпульсів ведучого вузла і входна лінія тактових імпульсів веденого вузла. Лінія SCK використовується для синхронізації передачі даних між ведучим і веденим інтерфейсами по лініях MOSI і MISO.

Приклад застосування інтерфейсу подано на рис. 3.13:

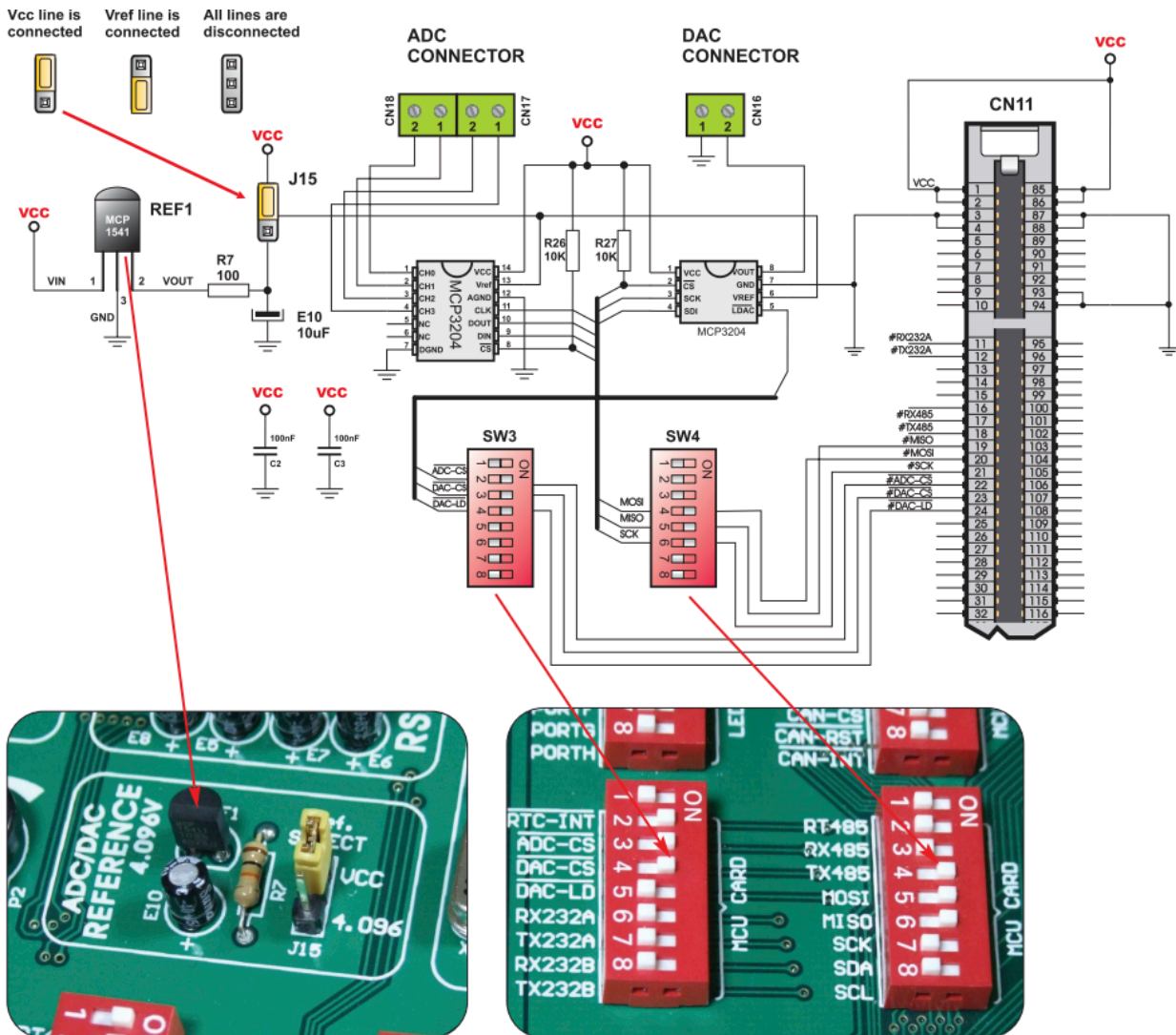


Рисунок 3.13 – Інтерфейс SPI на стенді UNI-DS3 [8]

I2C (Inter-Integrated Circuit)

Це – інтерфейс для підключення сенсорів та інших компонентів з низькими вимогами до швидкості передачі даних. I2C є послідовним синхронним інтерфейсом по двох провідниках, розроблений фірмою Philips. Він призначений для обміну інформацією між контролером та периферійними пристроями на невеликій відстані на швидкостях до 400 кбіт/с.

Переважно I2C використовується у вбудованих системах для керування та зв'язку з різними датчиками, годинниками реального часу, EEPROM, SDRAM, дисплеями та ін. пристроями, які знаходяться на тій же платі.

Дані в інтерфейсі I2C передаються по шині, яка складається з двох проводів – провід даних - SDA (англ. serial data line) і провід тактування – SCL (англ. serial clock line). Також між пристроями є третій провідник – земля (GND) (рис. 3.13).

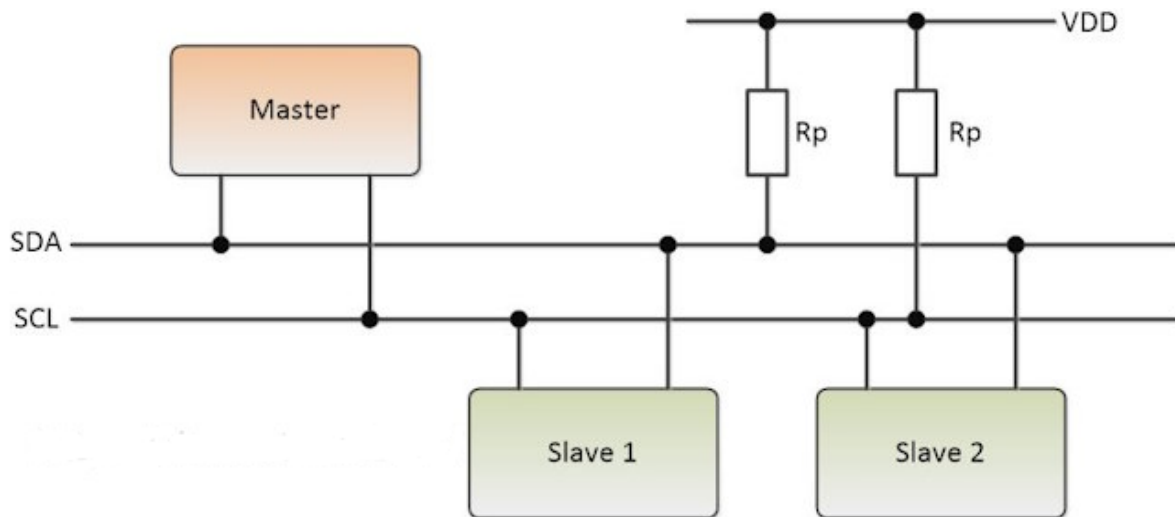


Рисунок 3.13 – Схема інтерфейсу I2C [8]

На шині можуть бути присутні головний пристрій, який називається master (ведучий, контролер) та декілька підлеглих пристроїв, які називаються slave (ведений, периферійний пристрій) [7].

UART (Universal Asynchronous Receiver-Transmitter)

Це простий інтерфейс для послідовної передачі даних між мікроконтролером та периферійними пристроями. Він використовується для підключення до модулів GPS, GSM, Bluetooth, Wi-Fi та працює за принципом передачі даних по двох проводах TX та RX, підходить для низькошвидкісної передачі даних, не вимагає синхронізації годинникових сигналів.

UART є протоколом асинхронного приймача/передавача і визначає набір правил для обміну послідовними даними між двома пристроями. В ньому дані передаються лише по двох дротах (є ще провід землі). Асинхронний означає відсутність тактового сигналу для синхронізації вихідних бітів від передавального пристрою до приймального кінця.

Інтерфейс UART може використовуватись для обміну даними між двома мікроконтролерами. Він широко поширений в електронці, його у різних видах використовують і великі та малі комп'ютери, контролери, датчики, засоби комунікації та інші електронні пристрої.

В цьому протоколі дані передаються послідовно (у вигляді бітів логічних нулів та одиниць) та одиниця відрізняється від нуля виключно за часом між перепадами рівнів сигналу, який заздалегідь задається швидкістю передачі (рис. 3.14).

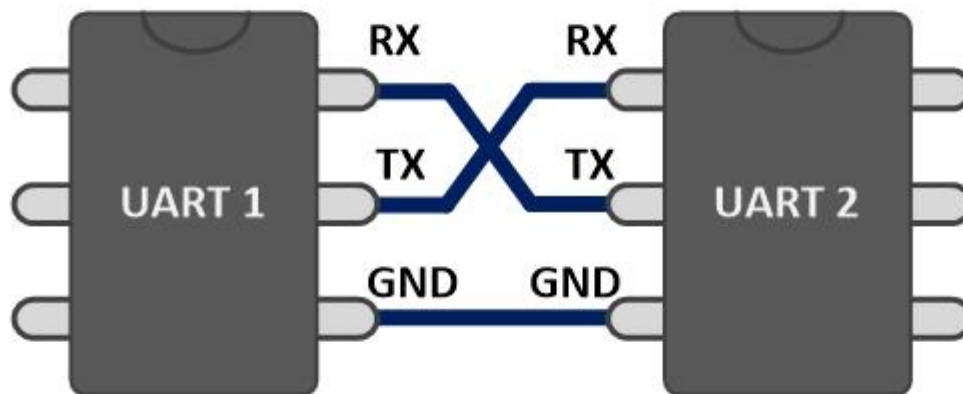


Рисунок 3.14 – Схема протоколу UART [8]

Комунікація в UART може бути симплексною (дані надсилаються лише в одному напрямку), напівдуплексною (кожна сторона передає тільки в один проміжок часу) або повнодуплексною (обидві сторони можуть передавати одночасно).

Периферійні інтерфейси відіграють ключову роль у функціонуванні IoT-пристроїв, оскільки вони забезпечують ефективну взаємодію мікроконтролерів із сенсорами, модулями зв'язку та іншими периферійними компонентами. Вибір інтерфейсу залежить від специфічних вимог до швидкості передачі даних, енергоефективності, вартості і складності реалізації.

Контрольні питання:

1. У чому полягає суть принципів архітектури Джона фон Неймана?
2. Що таке принцип двійкового кодування і чому він є основою для роботи комп'ютерів?
3. Як реалізується принцип програмного керування у комп'ютерній архітектурі?
4. Що означає принцип однорідності пам'яті у фон-нейманівській архітектурі?
5. У чому полягає принцип адресності і як він впливає на організацію пам'яті?
6. Які ключові відмінності між фон-нейманівською та гарвардською архітектурами?
7. Які переваги має гарвардська архітектура у порівнянні з фон-нейманівською?
8. Які існують основні особливості архітектур мікропроцесорів, що використовуються для збору, обробки та передачі даних?
9. Чим відрізняються архітектури процесорів, призначених для IoT-пристроїв, від традиційних ПК-процесорів?
10. Які існують типи комп'ютерних інтерфейсів за способом передачі даних?
11. Що таке послідовний та паралельний інтерфейси і які їхні основні відмінності?

ТЕМА 4

ПАМ'ЯТЬ ТА ІЄРАРХІЯ ПАМ'ЯТІ

Ієрархія пам'яті та будова пам'яті комп'ютера. Пам'ять з довільним та послідовним доступом. Кеш нульового рівня як особливість високопродуктивних систем.

Ієрархія пам'яті та будова пам'яті комп'ютера.

Ієрархія пам'яті відноситься до структури зберігання та доступу до даних у комп'ютерній системі. Вона складається з різних рівнів пам'яті, кожен з яких має різні швидкості, ємності та витрати (рис. 4.1).



Рисунок 4.1 – Піраміда ієрархії пам'яті комп'ютера [6]

Основна мета ієрархії пам'яті – забезпечити якнайшвидший доступ до найчастіше використовуваних даних.

Ієрархія пам'яті спроектована для оптимізації часу доступу до даних, розміщуючи найчастіше використовувані дані на найшвидших рівнях пам'яті. Розглянемо різні рівні пам'яті в ієрархії:

Регістри

На верхньому рівні ієрархії пам'яті знаходяться регістри процесора. Регістри - це найшвидший тип пам'яті, розташований всередині процесора. Вони зберігають дані, які в даний момент обробляє процесор. Регістри зберігають невелику кількість даних безпосередньо на процесорі, що дозволяє швидкий доступ і обробку. Однак, їхня ємність дуже обмежена, зазвичай у діапазоні кількох кілобайт.

Кеш-пам'ять

Наступний рівень ієрархії пам'яті - це кеш-пам'ять. Кеші - це маленькі, високошвидкісні блоки пам'яті, які зберігають часто запитувані дані. Вони заповнюють розрив у швидкості між процесором та основною пам'яттю (ОЗП). Кеш-пам'ять призначена для зберігання копій даних з основної пам'яті, які процесор, можливо, скоро буде запитувати. Завдяки розташуванню даних ближче до процесора кеш-пам'ять знижує час, необхідний для отримання часто запитуваної інформації.

Кеші зазвичай поділяються на кілька рівнів: L1, L2 і іноді навіть L3. Кеш L1 є найменшим, але найшвидшим, тоді як кеш L3 є найбільшим, але повільнішим у порівнянні з L1 і L2.

Основна пам'ять (ОЗП)

Основна пам'ять, часто звана ОЗП (оперативна запам'ятовуюча пам'ять), є основним сховищем, яке використовує комп'ютер. Вона є більшою у порівнянні з кеш-пам'яттю і зберігає програмні інструкції та дані, над якими в даний момент працює комп'ютер. Основна пам'ять забезпечує швидший доступ у порівнянні з вторинним сховищем. Однак, вона повільніша за реєстри та кеші. ОЗП є енергонезалежною, що означає, що її вміст втрачається під час вимикання або перезавантаження комп'ютера.

Розмір ОЗП зазвичай становить від кількох гігабайт до кількох терабайт у високопродуктивних серверах.

Вторинне сховище

Вторинне сховище відноситься до пристроїв зберігання таких, як жорсткі диски, твердотільні накопичувачі (SSD) та інші енергонезалежні пристрої зберігання. Воно забезпечує набагато більші ємності зберігання у порівнянні з основною пам'яттю, але працює повільніше за швидкістю доступу. Вторинне сховище використовується для довготривалого зберігання даних, таких як встановлені операційні системи, прикладне програмне забезпечення, документи та медіафайли. У порівнянні з ОЗП, вторинне сховище значно повільніше, але пропонує набагато більшу ємність зберігання. Жорсткі диски (HDD) є типом вторинного сховища, а SSD забезпечують швидший доступ, але за вищою ціною.

Третинне сховище

Найнижчий рівень ієрархії пам'яті - це третинне сховище, яке включає оффлайн-пристрої зберігання, такі як оптичні диски та магнітні стрічки. Третинне сховище має найбільші ємності, але є набагато повільнішим у порівнянні з іншими типами пам'яті. Ці пристрої зберігання зазвичай використовуються для довготривалого резервного копіювання і архівування даних, де швидкість не є основним пріоритетом. Третинне сховище часто звертають на нього рідко і вимагає ручного втручання для отримання даних.

Ієрархія пам'яті відіграє вирішальну роль у продуктивності та ефективності комп'ютерних систем. Розміщуючи часто використовувані дані на швидших рівнях пам'яті, вона оптимізує час доступу до даних, покращуючи загальну чутливість системи.

Наведемо приклади практичних застосувань, де ієрархія пам'яті є важливою:

Кешування даних. Техніки кешування використовуються для зменшення часу доступу до часто використовуваних даних. Кеші призначені для зберігання копій даних, які найближчим часом будуть запитуватися, зменшуючи потребу в отриманні даних з повільніших рівнів ієрархії пам'яті.

Оптимізація алгоритмів та програмного забезпечення. Ефективні алгоритми та практики кодування можуть мінімізувати потребу в надмірному доступі до даних, знижуючи навантаження на ресурси пам'яті. Завдяки розробці алгоритмів, що мінімізують операції з пам'яттю та максимізують локальність даних, можна додатково покращити продуктивність ієрархії пам'яті.

Оновлення апаратного забезпечення. Регулярне оновлення апаратного забезпечення, особливо основної пам'яті (ОЗП), дозволяє комп'ютерам відповідати зростаючим вимогам до обробки даних. Збільшення ємності ОЗП може значно зменшити потребу в доступі до повільнішої вторинної або третинної пам'яті.

Комп'ютерна пам'ять забезпечує підтримку однієї з найважливіших функцій сучасного комп'ютера – здатність тривалого зберігання інформації. Центральний процесор і пристрій, що запам'ятовує, є ключовими ланками класичної архітектури фон Неймана – принципу, закладеного в основу більшості сучасних комп'ютерів загального призначення.

Основну пам'ять утворюють запам'ятовуючі пристрої з довільним доступом. Такі ЗП утворені як масив комірок, а «довільний доступ» означає, що звернення до будь-якої комірки займає один і той же час і може проводитися в довільній послідовності. Кожна комірка містить фіксоване число запам'ятовуваних елементів і має унікальну адресу, що дозволяє розрізняти комірки під час звернення до них для виконання операцій запису і зчитування.

Будова пам'яті комп'ютера

Центральний процесор може безпосередньо звертатися лише до основної пам'яті (оперативної пам'яті, ОЗП), за винятком власних регістрів. Дані, що знаходяться на зовнішніх запам'ятовуваних пристроях, стають доступними для процесора тільки після їхнього завантаження в оперативну пам'ять.

У англійській літературі ОЗП відповідає аббревіатура **RAM** – Random Access Memory, тобто «пам'ять з довільним доступом», що не зовсім коректно, оскільки пам'яттю з довільним доступом є також ПЗП і регістри процесора. Для більшості типів напівпровідникових ОЗП характерна енергозалежність – навіть під час 1 Вольт-секундного переривання живлення інформація, що зберігається, втрачається. Мікросхема ОЗП повинна бути постійно підключена до джерела живлення і тому може використовуватися тільки як тимчасова пам'ять. Другу групу напівпровідникових ЗП основної пам'яті утворюють незалежні мікросхеми ПЗП (**ROM** – Read-Only Memory). ПЗП забезпечує зчитування інформації, але не допускає її зміни (у ряді випадків інформація в ПЗП може бути змінена, але цей процес сильно відрізняється від зчитування і вимагає значно більшого часу).

Для того, щоб детально уявити будову оперативної пам'яті комп'ютера, уявіть собі Excel. Сторінка в Excel складається з клітинок (комірок), і кожна комірка має свій унікальний номер (A1, A2, ... B1, B2). Якщо знати номер комірки, завжди можна записати в неї якесь значення або ж отримати значення, що там зберігається. Пам'ять комп'ютера побудована за схожим принципом. Тобто, комірки пам'яті мають адресу (порядковий номер, число), за якою до них можуть звертатися команди процесора.

Під час роботи програма та її дані зберігаються в пам'яті. Уся пам'ять комп'ютера складається з маленьких комірок – байтів. Кожна комірка має свій унікальний номер – 0, 1, 2, 3, ...; (нумерація починається з нуля). Знаючи номер комірки, ми зможемо зберегти в ній певні дані. Або взяти їх із цієї комірки.

Процесор уміє виконувати команди із завантаженої в пам'ять програми.

Майже всі команди процесора – це щось на кшталт узяти з деяких комірок дані → щось із ними зробити → результат помістити в інші комірки

В одних комітках зберігається код програми – набір команд для процесора, в інших – дані цієї програми. Номер комірки також називають *адресою* комірки.

Одним із ключових понять роботи комп'ютера також є поняття *адресного простору* процесора. Адресний простір процесора позначає діапазон можливих адрес, які він може використовувати при завантаженні та зберіганні в пам'ять. Не слід думати, що терміни «адресний простір» і «оперативна пам'ять» еквівалентні. Адресний простір – це просто набір адрес, які уміє формувати процесор і зовсім не обов'язково всі ці адреси відповідають реально існуючим елементам пам'яті. Залежно від модифікації персонального комп'ютера і складу його периферійного устаткування, розподіл адресного простору може дещо відрізнятися.

Пам'ять з довільним та послідовним доступом.

Ємність основної пам'яті сучасних комп'ютерів достатньо велика, щоб її можна було реалізувати на одній інтегральній мікросхемі (ІМС). Необхідність об'єднання декількох ІМС виникає також, коли розрядність комірок у мікросхемі запам'ятовуючого пристрою менше розрядності слів комп'ютера.

Збільшення розрядності Запам'ятовуючих пристроїв (ЗП) реалізується за рахунок об'єднання адресних входів ІМС ЗП, які об'єднуються. Інформаційні входи і виходи мікросхем є входами і виходами модуля ЗП збільшеної розрядності (рис. 4.2).

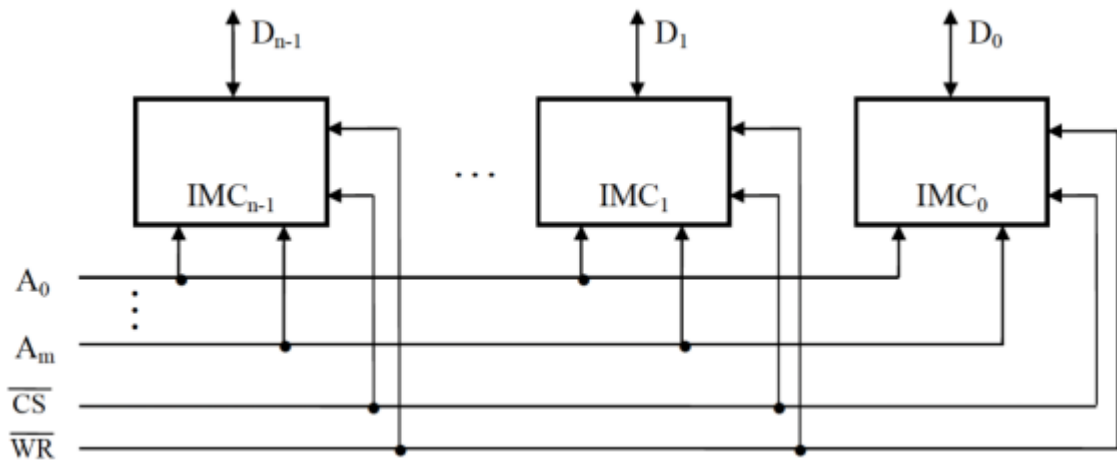


Рисунок 4.2 – Збільшення розрядності пам'яті [6]

Таку об'єднану сукупність мікросхем називають модулем пам'яті. Модулем можна називати і одну мікросхему, якщо вона вже має необхідну розрядність. Один або декілька модулів утворюють банк пам'яті.

Для отримання потрібної ємності ЗП потрібно певним чином об'єднати декілька банків пам'яті меншої ємності. Взагалі основна пам'ять комп'ютера практично завжди має блочну структуру, тобто містить декілька банків. Під час використання блочної пам'яті, яка складається з B банків, адреса комірки A перетворюється в пару (b, w) , де b - номер банку, w - адреса комірки всередині банку. Відомі три схеми розподілу розрядів адреси A між b і w : блочна; циклічна; блочно-циклічна. Типова структура блочної пам'яті показана на рис. 4.3.

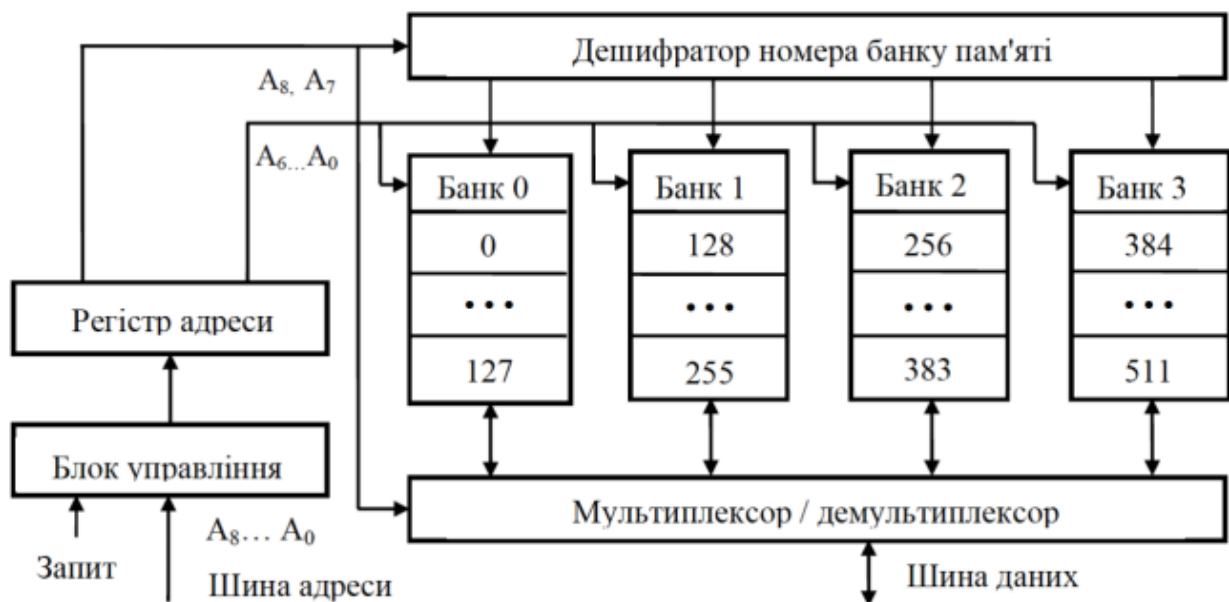


Рисунок 4.3 – Блочна структура пам'яті комп'ютера [6]

У функціональному відношенні така пам'ять може розглядатись як єдиний ЗП, ємність якого дорівнює сумі ємностей складових, а швидкодія – швидкодії окремого банку. Блочний принцип побудови оперативної пам'яті має ще одну перевагу – дозволяє зменшити час доступу до інформації. Це стає

можливим завдяки потенціальному паралелізму, який є властивим для блочної організації. Більшої швидкості доступу можна досягти за рахунок одночасного доступу до багатьох банків пам'яті.

Одна з використовуваних для цього методик називається *розширенням пам'яті*. В її основу покладене так зване чергування адрес (address interleaving – інтерлів), яке полягає в зміні системи розподілу адрес між банками пам'яті. Причому чергування адрес базується на властивості локальності по зверненню, відповідно до якого послідовний доступ у пам'ять звичайно виконується до комірок, які мають суміжні адреси. Іншими словами, якщо в даний момент виконується звернення до комірки з адресою 5, то наступне звернення найвірогідніше буде до комірки з адресою 6, а потім 7 і т.д [6].

Чергування адрес забезпечується за рахунок циклічного розбиття адреси (для вибору банку використовуються два розряди адреси, а інші розряди – для вибору комірки в банку). У пам'яті з інтерлівом кожен наступний рядок вибирається в новому банку, що дозволяє запам'ятовуваним елементам рядка з попереднього банку відновлювати свій стан (рис. 4.4).

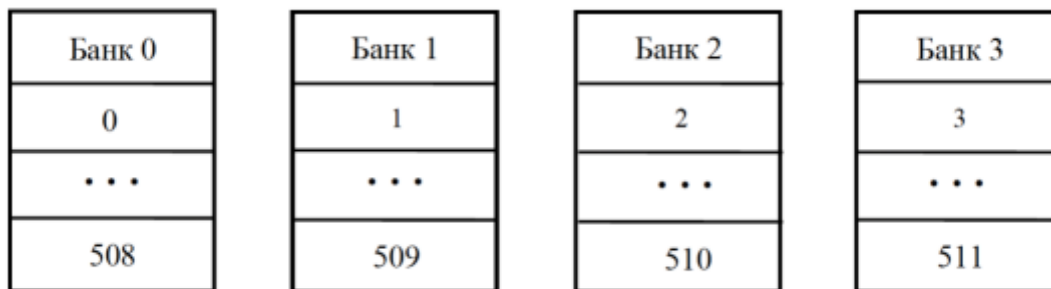


Рисунок 4.4 – Нумерація комірок пам'яті комп'ютера [6]

У разі великої кількості банків середній час доступу до основної пам'яті скорочується в N разів (N - кількість банків), але за умови, що комірки, до яких проводиться звернення, відносяться до різних банків. Якщо ж запити до одного й того ж банку приходять один за одним, то кожний наступний запит повинен очікувати завершення обслуговування попереднього. Така ситуація називається *конфліктом по доступу*. У разі частого виникнення конфліктів по доступу метод стає неефективним.

У блочно-циклічній схемі розширення пам'яті кожний банк складається з декількох модулів, що адресуються за круговою схемою (рис. 4.5). Адреси між банками розподілені по блочній схемі. Таким чином, адреса комірки розбивається на три частини. Старші біти визначають номер банку, наступна група розрядів адреси вказує на комірку в модулі, а молодші біти адреси вибирають модуль у банку.

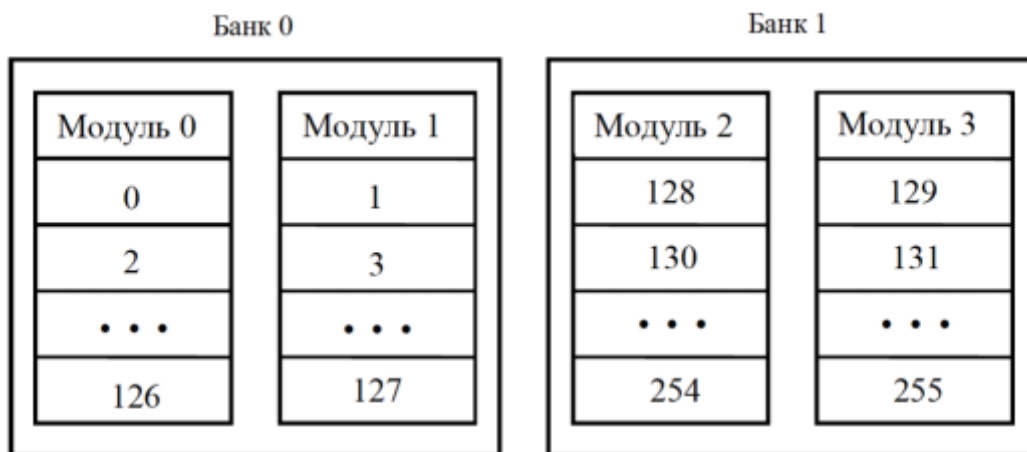


Рисунок 4.5 – Блочно-циклічна схема розшарування пам'яті [6]

У багатопроцесорних системах з загальною пам'яттю, де запити на доступ до пам'яті достатньо незалежні, у систему включають декілька контролерів пам'яті, що дозволяє окремим банкам функціонувати абсолютно автономно.

Як оперативну пам'ять комп'ютера використовують динамічну пам'ять, яка здатна зберігати інформацію тільки протягом достатньо короткого проміжку часу, після якого інформацію необхідно відновлювати, інакше вона буде загублена. Аббревіатура динамічної пам'яті – DRAM (Dynamic Random Access Memory). Динамічна пам'ять здатна запам'ятовувати біти інформації завдяки паразитній ємності (близько 10-15 Ф). Ця пам'ять сконструйована на базі структури компліментарної технології метал-оксид-напівпровідник (CMOS – Complimentary Metal Oxide Semiconductor). За CMOS-технологією, завдяки її безсумнівним технічним перевагам, будуються сучасні чіпи швидкодіючих електронних елементів з високою щільністю упакування [6].

Кеш нульового рівня як особливість високопродуктивних систем

Кеш – це швидша і менша за розміром пам'ять, яка зберігає копії даних, що часто використовуються, з головної пам'яті.

Питання ієрархії кеш-пам'яті, термінологію «промах кешу», «попадання в кеш», асоціативність кешу розглянута в курсі вивчення дисципліни на лабораторних заняттях та самостійній роботі.

Кеш-пам'ять процесора – це дуже швидка пам'ять, яка відіграє роль буфера між процесором (CPU) та оперативною пам'яттю (RAM). Кеш-пам'ять зберігає дані та інструкції, до яких звертаються найчастіше, завдяки чому процесор може отримувати доступ до всього цього практично миттєво.

Важливо, що у процесорах є особлива пам'ять, представлена регістрами процесора, яка зазвичай є невеликим сховищем інформації, що забезпечує вкрай високу швидкість обміну даними. Регістри - це найшвидша пам'ять, з якою може працювати процесор, яка розташована максимально близько до інших механізмів і має невеликий обсяг (про це сказано у темі №1 лекційного матеріалу).

Проте, іноді регістри називають кешем нульового рівня (L0 Cache, L – це скорочення від Layer). У процесорів, крім того, є доступ до кількох рівнів кеш-пам'яті. Це – до чотирьох рівнів кешу, які, відповідно, називаються кешами першого, другого, третього та четвертого рівня (L0-L4 Cache).

Те, до якого рівня відносяться регістри процесора, зокрема, це кеш нульового чи першого рівня, визначається архітектурою процесора і материнської плати. Крім того, від архітектури системи залежить те, де саме – на процесорі, чи материнській платі, фізично розташована кеш-пам'ять різних рівнів.

Кеш L0 є рідкісною і специфічною функцією, яка зустрічається в деяких високопродуктивних системах, таких як графічні процесори (GPU), або в процесорах, що працюють з великими обсягами даних. Він грає роль ще швидшого буфера, ніж кеш першого рівня (L1), що дозволяє суттєво збільшити ефективність роботи обчислювальних блоків.

Особливості кешу нульового рівня в високопродуктивних системах забезпечується надшвидким доступом до даних. Це пов'язано з тим, що кеш L0 розташований дуже близько до обчислювальних блоків, таких як ядра процесора або графічні ядра, що дозволяє їм отримувати дані майже миттєво. Це забезпечує мінімальні затримки під час виконання критично важливих обчислювальних операцій. Він володіє малим обсягом (від кількох кілобайт до кількох десятків кілобайт) та високою швидкістю. Його невеликий розмір дозволяє забезпечити надзвичайно низьку латентність (затримку) у порівнянні з більшими кешами вищих рівнів (L1, L2, L3), які можуть мати більшу ємність, але нижчу швидкість.

L0 актуальний для графічних процесорів (GPU) та часто використовується для оптимізації обчислень при рендерингу графіки або виконанні паралельних обчислювальних задач, як у випадку архітектури AMD RDNA 2 та 3, де L0 кеш прив'язаний до WGP (Workgroup Processor). GPU використовують кеш L0 для зменшення частоти звернення до повільнішої пам'яті, такої як оперативна пам'ять або кеші наступних рівнів.

У високопродуктивних обчислювальних системах, які виконують паралельні завдання (наприклад, обробка потоків даних або штучний інтелект), кеш L0 може використовуватись для швидкої координації між ядрами або блоками. Це особливо корисно для роботи із великими масивами даних, що потребують швидкого доступу. Завдяки швидшому доступу до необхідних даних, кеш L0 допомагає знизити енергоспоживання, оскільки процесору не потрібно звертатися до кешу вищих рівнів або до оперативної пам'яті, що займає більше часу і енергії.

У спеціалізованих обчислювальних системах для вузького кола завдань, таких як обробка графіки, моделювання, машинне навчання та високопродуктивні обчислення (HPC) використання L0 дозволяє оптимізувати конкретні обчислювальні операції, які вимагають максимальних швидкостей доступу до даних.

Контрольні питання:

1. Що таке ієрархія пам'яті в комп'ютерній архітектурі і яка її мета?
2. Які рівні пам'яті входять до типової ієрархії комп'ютера?
3. У чому полягає різниця між оперативною пам'яттю (RAM) та постійною пам'яттю (ROM)?
4. Що таке пам'ять з довільним доступом (RAM) і як вона працює?
5. У чому полягає принцип роботи пам'яті з послідовним доступом?
6. Яку роль у комп'ютері відіграє кеш-пам'ять і скільки рівнів кешу зазвичай використовуються?
7. Що таке кеш нульового рівня (L0) і в яких системах він застосовується?
8. Чим відрізняється кеш L0 від кешів L1, L2 і L3?
9. Як використання багаторівневої кеш-пам'яті впливає на продуктивність системи?
10. Яким чином організовано взаємодію між центральним процесором та різними рівнями пам'яті?

ТЕМА 5. БУДОВА ПРОЦЕСОРА І РЕГІСТРОВОЇ ПАМ'ЯТІ

Внутрішня будова процесора. Архітектура x86. Надоперативна регістрова пам'ять. Регістри загального призначення, спеціального призначення, вказівник стека, базовий та індексний регістри. Забезпечення доступу до критичних даних і інструкцій, необхідних для роботи процесора.

Внутрішня будова процесора

Центральний процесор є основним і найголовнішим елементом системи. Завдяки йому виконуються всі завдання пов'язані з передачею даних, виконанням команд, логічними і арифметичними діями. Перед тим, як розібрати основні принципи роботи CPU, бажано ознайомитися з його компонентами, адже це не просто прямокутна пластина, вмонтовується в материнську плату, це складний пристрій, що утворюється з багатьох елементів (рис. 5.1).

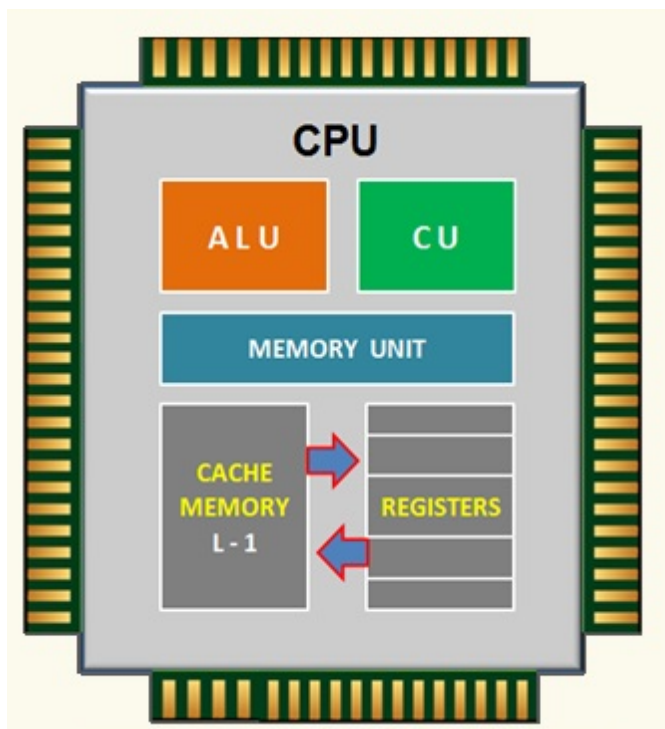


Рисунок 5.1 – Основні структурні блоки процесора [9]

Блок керування (CU) керує виконанням інструкцій, координує потік даних і визначає послідовність операцій.

Арифметико-логічний блок (ALU) виконує арифметичні обчислення та логічні операції, обробляючи основні математичні обчислення.

Блок пам'яті тимчасово зберігає дані та інструкції, до яких часто звертаються, скорочуючи час пошуку та покращуючи загальну продуктивність.

Блок ресурсів керує основними ресурсами центрального процесора, такими як регістри та шини, сприяючи ефективному обміну даними між різними компонентами.

Процесор інтерпретує та виконує інструкції, виконує обчислення та координує різні завдання в системі. Без центрального процесора комп'ютер не міг би ефективно функціонувати або виконувати будь-які завдання обробки.

Архітектура x86

Найпопулярніша (навіть у 2025 році) десктопна архітектура x86 бере свій початок у 1978 році, коли компанія Intel випустила процесор Intel 8086. Саме його номер став основою для назви всієї архітектури. Процесор був 16-бітним, працював на частоті 5–10 МГц і підтримував до 1 мегабайта пам'яті. Він заклав основні принципи побудови процесорів: використання сегментованої пам'яті, набір складних інструкцій (CISC) та базовий набір регістрів загального призначення. У 1982 році з'явився процесор Intel 80286, який удосконалив ідеї 8086, додавши захищений режим роботи, що дозволило краще працювати з пам'яттю і підвищити безпеку програм.

У 1985 році Intel представила процесор 80386, який став справжнім проривом – архітектура перейшла до 32-бітної розрядності. Це значно збільшило можливості для обробки даних і використання пам'яті.

Далі, у 1989–1993 роках, виходять процесори Intel 80486 і перші серії Pentium, де з'являються вбудовані арифметичні співпроцесори та починається використання конвеєризації для пришвидшення виконання команд. Саме на Pentium архітектура x86 почала активно завойовувати масовий ринок домашніх і офісних комп'ютерів.

На початку 2000-х років з'являються великі зміни: AMD випускає Athlon64 і впроваджує 64-бітне розширення архітектури x86, відоме як x86-64 або просто x64. Intel теж переймає цей стандарт у своїх процесорах. Це дозволяє працювати з великими обсягами оперативної пам'яті й запускати сучасні операційні системи. Сьогодні практично всі популярні процесори Intel Core (i3, i5, i7, i9) і AMD Ryzen базуються на розширеній x86-архітектурі. Вона постійно розвивається, отримуючи нові набори інструкцій для оптимізації мультимедійної обробки, штучного інтелекту, шифрування й інших сучасних завдань, і залишається домінуючою в світі персональних комп'ютерів навіть через понад 45 років після свого створення.

Розглянемо схему структури процесора архітектури x86, а саме класичної архітектури процесорів Intel 8086 / 8088. Вона показує основні компоненти і їхню взаємодію всередині центрального процесора (ЦП) (рис. 5.2)

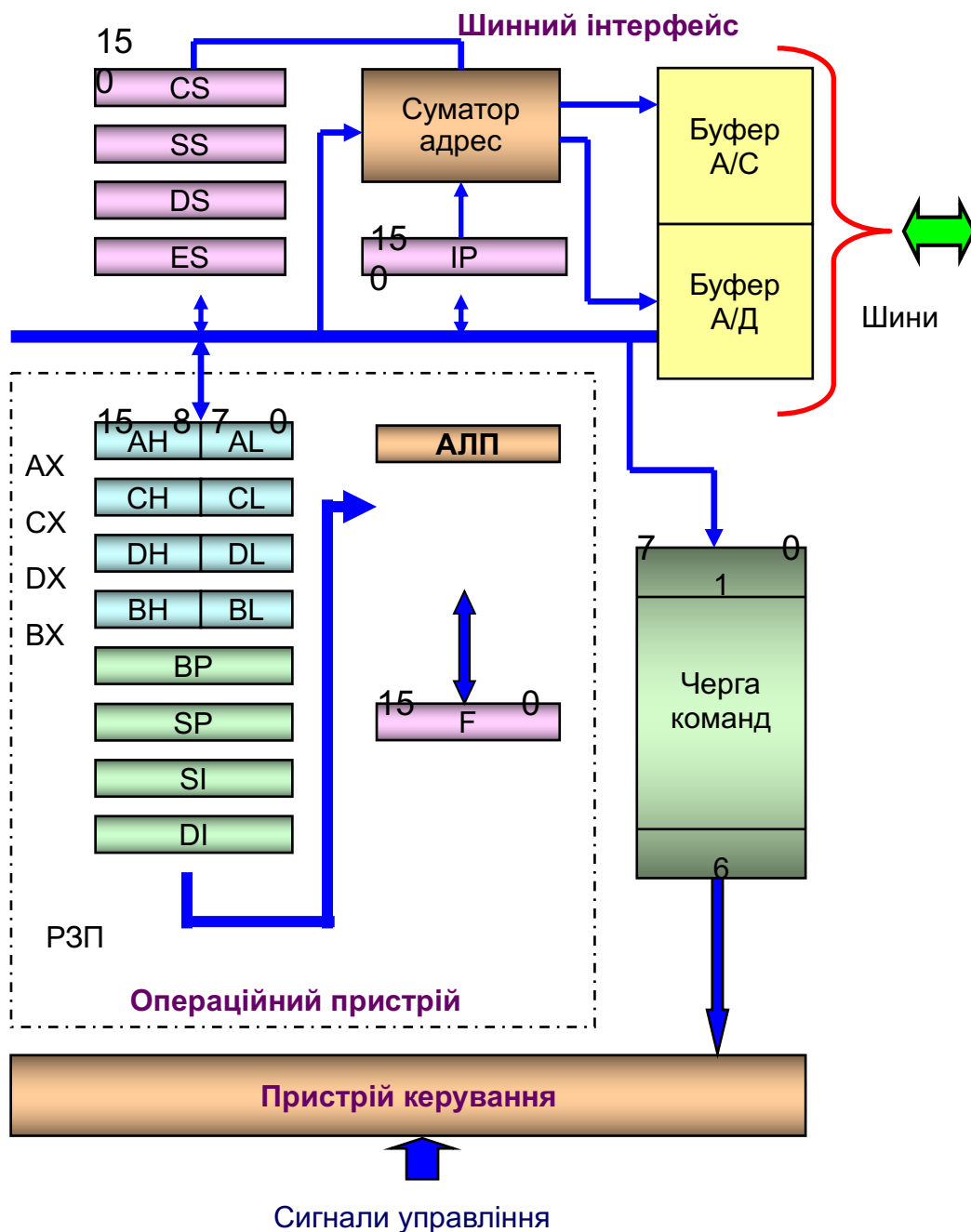


Рисунок 5.2 – Структура класичного x86 процесора [10]

Шинний інтерфейс – забезпечує обмін даними і адресами між процесором і пам'яттю чи пристроями введення-виведення через буфери А/С (Адреса/Керування) і А/Д (Адреса/Дані).

Суматор адрес і ІР (сегментний показник команд) – формують фізичну адресу для доступу до пам'яті.

Операційний пристрій – це набір регістрових пам'ятей: регістрів загального призначення (АВ, АЛ, СВ, СЛ, ДВ, ДЛ, ВВ, ВЛ), регістрів вказівників (ВР, СР) і регістрів індексів (SІ, ДІ).

АЛП (арифметико-логічний пристрій) виконує арифметичні та логічні операції над даними з регістрів. F (регістр прапорців) – зберігає результати операцій АЛП (наприклад, ознака переносу, нульовий результат тощо).

Черга команд виконує попереднє завантаження інструкцій для підвищення швидкодії за принципом конвеєра.

Пристрій керування генерує сигнали управління для всіх блоків процесора, забезпечуючи їхню скоординовану роботу.

Ця схема демонструє розділення процесора на дві основні частини: операційний пристрій для роботи з даними і пристрій керування для організації виконання команд.

Надоперативна регістрова пам'ять. Регістри загального призначення, спеціального призначення, вказівник стека, базовий та індексний регістри.

Регістри – це високошвидкісні сховища даних, розташовані безпосередньо на процесорі. Вони використовуються для тимчасового зберігання інструкцій, даних та адрес пам'яті під час виконання програм. Регістри дозволяють швидко отримувати доступ до необхідних даних, що значно підвищує швидкодію процесора. Існують різні типи регістрів, такі як регістр інструкцій (Instruction Register), регістр програмного лічильника (Program Counter) та регістри загального призначення, які виконують різні функції під час роботи процесора та інші.

Усі регістри процесора можна об'єднати в групи за схожістю виконуваних ними функцій.

У першу групу, названу групою **регістрів загального призначення**, входять регістри AX, BX, CX, DX. Вони призначені в основному для збереження даних. Регістри BX і DX можна додатково використовувати як адресні: регістр BX – як адреса зміщення байта чи слова в оперативній пам'яті, регістр DX – як адреса порту вводу-виводу. Кожен з цих регістрів має свої особливості. Наприклад, регістр AX завжди є одним з операндів команди множення, регістр CX використовується у якості лічильника у команді LOOP, DX – як розширення регістра AX у командах множення і ділення. Ці регістри є 16-розрядними, і складаються з двох однобайтових регістрів кожен: AX – з AH і AL, BX – з BH і BL і т.д.

Існує вісім 16-бітових робочих регістрів. Це AX, BX, CX, DX, SI, DI, BP і SP. Будь-які з них можуть бути задіяний в обчислювальному процесі, але необхідно враховувати, що все ж таки багато інструкцій процесора для роботи вимагають використання строго певних регістрів з цієї групи.

Регістр AX (Accumulator) доцільно використовувати для тимчасового зберігання даних (наприклад в обчисленнях); BX (Base) – для завдання адрес в пам'яті; CX (Count) – для рахунку (кількість повторів циклу); DX (Data) – для зберігання даних в деяких арифметичних операціях а також адрес для доступу до шини введення-виведення.

Регістр BP (Base Pointer) схожий з регістром BX, але в основному використовується для доступу до передаваних через стек параметрів з підпрограми.

Серед всіх регістрів виділяється SP (Stack Pointer), за допомогою якого задається адреса верхівки програмного стека. Це дуже важливо у багатьох випадках, тому строго не рекомендується використовувати SP в іншій ролі.

Окрім 16-бітових регістрів, в процесорі 8086 присутні ще вісім 8-бітових регістрів – AL, AH, BL, BH, CL, CH, DL і DH. Але ці регістри не реалізовані окремо від 16-бітових на фізичному рівні. Кожний з них – це певна частина 16-бітового регістра. Наприклад, AL означає «Ax Low» – молодший байт AX, а CH – «Cx High» – старший байт CX. Важливо запам'ятати, що зміна будь-якого з 8-бітових регістрів спричинить за собою зміну і відповідного йому 16-бітового регістра, як і навпаки.

В розглянутому на попередньому рисунку процесорі 8086 існує два **регістри спеціального призначення** – IP (Instruction Pointer) і FLAGS. Процесор особливим чином маніпулює цими регістрами, тому до них немає прямого доступу, як до РЗП або сегментним регістрам.

IP – це 16-бітовий регістр, який використовується для завдання адреси інструкції процесора, яка виконується в даний момент часу.

Регістр FLAGS не схожий на решту регістрів процесора – він не береже числових значень, а є «набором» однобітових значень – прапорців, які характеризують стан процесора. Хоч FLAGS має довжину 16 біт, реально використовуються тільки 9 з них. Прапорці, що часто використовуються, – zero (нуль), carry (перенесення), sign (знак) і overflow (переповнювання). Це так звані прапорці умови.

Оперативна пам'ять комп'ютера складається з осередків розміром в один байт. Загальна кількість байт $= 2$ в ступені ША. Процесор 8086 16-розрядний, але ША 20-розрядна, т.о. можна адресувати 1 Мбайт пам'яті. Це досягається завдяки сегментації - розбиттям пам'яті на блоки завдовжки в 64 Кілобайти, так звані сегменти. Положення сегменту в пам'яті описується 16-бітовим значенням, яке, для отримання доступу до цього сегменту, повинне зберігатися в одному з сегментних регістрів. А усередині кожного з сегментів дані адресуються через 16- розрядний зсув (offset) щодо початку сегменту.

снують чотири сегментні регістри - це CS (Code Segment), DS (Data Segment), ES (Extra Segment) і SS (Stack Segment). Всі ці регістри можуть використовуватися для роботи з будь-якими сегментами пам'яті, але є деякі обмеження.

CS повинен бути пов'язаний з сегментом, який береже виконувани в даний момент команди процесора (code). Щоб виконати інструкції, що знаходяться в інших сегментах, просто необхідно змінити вміст CS.

Регістр SS використовується для операцій із стеком програми. Стек працює за принципом FILO (First Input Last Output). Це означає, що при читанні ми забираємо із стека останнє записане в нього дане. В більшості випадків стек використовується для збереження стану процесора (наприклад,

вмісту регістрів), передачі параметрів в підпрограми і ін., тому змінювати SS рекомендується тільки в окремих випадках і одночасно з SP.

DS і ES можуть бути пов'язаний з будь-якими сегментами пам'яті, хоча зручніше всього було б DS застосовувати для доступу до даних програми (наприклад, змінним), що часто використовуються, а ES – як додатковий регістр, коли інші сегментні регістри задіяні, і небажано їх зміна.

Чотири **індексні регістри** мікропроцесора i386 можуть адресуватись як 32-бітні (EBP, ESI, EDI, ESP), а також як 16-бітні (BP, SI, DI, SP). Ці регістри використовують також як адресні, або як операнди в командах обробки даних.

Регістр ESP використовується у якості **вказівника вершини стека**.

Шість 16-розрядних сегментних регістрів (CS, SS, DS, ES, FS, GS) вказують на початок відповідного сегмента пам'яті. Призначення регістрів CS, SS, DS, ES та відповідних сегментів таке ж саме, як у процесорі x86. Сегментні регістри FS, GS використовуються як додаткові сегменти даних. Лічильник команд EIP має розрядність 32 біти та призначений для визначення адреси наступної команди. Вміст лічильника інкрементується при виконанні кожного байта команди. При виникненні переривань та виключень, а також при виконанні команд передачі керування вміст лічильника команд перезавантажується новим значенням, а черга команд скидається. Регістр ознак EFLAGS призначений для збереження ознак статусу та керування мікропроцесором. Ознаки поділяють на ознаки статусу та керування. Ознаки статусу визначають властивості результату виконання останньої арифметичної команди, команди маніпуляції бітами, або команди порівняння (рис. 5.3)

Ім'я	Назва	Функція
Арифметичні регістри (збереження даних)		
AX	акумулятор, accumulator register	усі чотири регістри допускають роздільне використання їх молодших байтів AL, BL, CL, DL і старших байтів AH, BH, CH, DH. Таким чином, забезпечується можливість обробки двобайтових слів та окремих байтів.
BX	базовий перістр, base register	
CX	лічильник, count register	
DX	перістр даних, data register	
Сегментні регістри (адресна інформація)		
CS	перістр кодового сегмента, code segment register	зберігають адреси відповідних сегментів
SS	перістр стекового сегмента, stack segment register	
DS	перістр сегмента даних, data segment register	
ES	перістр додаткового сегмента даних, extension data segment register	
Вказівні та індексні регістри (адресна інформація)		
BP	показчик бази, base pointer register	зберігають зсув адреси в межах поточного стекового сегмента пам'яті
SP	показчик стека, stack pointer register	
SI	індекс джерела, source index register	зберігають зсув адреси відповідно в поточному сегменті даних і в поточному додатковому сегменті
DI	індекс приймача, destination index register	
Регістри стану та управління		
F	перістр прапорців, Flag register	зберігає інформацію про стан як самого МП, так і програми, що виконується
IP	показчик команд, Instruction Pointer register	зберігає зсув наступної команди в поточному кодовому сегменті

Рисунок 5.3 – Функціональні особливості регістрів процесора архітектури x86 [10]

Забезпечення доступу до критичних даних і інструкцій, необхідних для роботи процесора.

Надоперативна регістрова пам'ять – це внутрішній набір регістрів у процесорі, який забезпечує надшвидкий доступ до критичних даних та інструкцій, необхідних для виконання команд.

Вона розташована безпосередньо в ядрі процесора й використовується для зберігання операндів, адрес та результатів обчислень, що дає змогу процесору працювати максимально ефективно, без затримок на звернення до кешу чи оперативної пам'яті.

Завдяки своїй швидкодії, регістрова пам'ять виконує ключову роль у забезпеченні миттєвого доступу до найважливіших даних, що критично важливо для швидкої та стабільної роботи арифметико-логічного пристрою, пристрою керування та всього процесора в цілому.

Забезпечення доступу до критичних даних і інструкцій, необхідних для роботи процесора, здійснюється через кілька основних механізмів, що стосуються організації пам'яті та доступу до неї.

Процесори використовують багаторівневу ієрархію пам'яті (регістра, кеш-пам'ять, основна пам'ять), щоб оптимізувати швидкість доступу до критичних даних та інструкцій. Інструкції та дані, що часто використовуються, зберігаються в кеш-пам'яті (L1, L2, L3), яка знаходиться ближче до процесора і має високий швидкість доступу. Всі ці дані і інструкції повинні бути наявні в кеші для забезпечення швидкого виконання команд процесора.

Процесор має власну набір реєстрів, які зберігають критичні дані, що необхідні для обчислень. Це може включати такі дані, як результати арифметичних операцій, адреси пам'яті, значення лічильників команд і інші важливі параметри.

Для організації доступу до пам'яті використовуються сегментація та сторінкова адресація, що дозволяє забезпечити ізоляцію та ефективне керування доступом до різних областей пам'яті, зокрема до критичних даних і інструкцій. Таблиці сторінок та переклад адреси на рівні апаратного забезпечення перекладають віртуальні адреси в фізичні, що дозволяє захистити важливі ділянки пам'яті від несанкціонованого доступу.

Процесор також використовує механізми захисту пам'яті, такі як різні рівні доступу до системи пам'яті (наприклад, ядро має більш високий рівень доступу, ніж користувацькі програми). Це дозволяє захистити критичні інструкції і дані, обмежуючи доступ до них. Додатково використовуються апаратні засоби для контролю доступу до пам'яті, такі як механізми NOEXEC (заборона виконання певних частин пам'яті) або DEP (Data Execution Prevention).

Контролери пам'яті управляють доступом до різних типів пам'яті (наприклад, RAM, ROM, кеш-пам'ять), забезпечуючи порядок і синхронізацію доступу до цих ресурсів, особливо при доступі до критичних інструкцій чи даних.

Контрольні питання:

1. Які основні компоненти складають внутрішню архітектуру процесора?
2. Як реалізована комунікація між арифметико-логічним пристроєм (АЛП) і регістрами процесора?
3. Які функції виконує контрольний пристрій у процесорі?
4. Як відбувається обробка інструкцій у процесорі?
5. Що таке шина даних і як вона впливає на внутрішню структуру процесора?
6. Які основні відмінності між архітектурами x86 та x86-64?
7. Як архітектура x86 організовує доступ до пам'яті за допомогою сегментів?
8. Що таке "режим реального часу" в процесорах x86 і в чому його основні обмеження?
9. Як працює режим захисту в архітектурі x86?
10. Що таке інструкційний конвеєр і як він реалізований у процесорах x86?
11. Що таке надоперативна пам'ять процесора і в чому її роль?
12. Для яких задач використовуються регістри загального призначення в процесорі?
13. Як регістри загального призначення взаємодіють з іншими компонентами процесора?
14. Як функціонують спеціальні регістри, такі як вказівник стека, базовий регістр та індексний регістр у контексті роботи процесора?

ТЕМА 6. ФУНКЦІЇ ТА ЕЛЕМЕНТАРНІ ОПЕРАЦІЇ АРИФМЕТИКО- ЛОГІЧНОГО ПРИСТРОЮ

Будова АЛП. Одноблокові та багатоблокові АЛП. Послідовні, послідовно-паралельні та паралельні АЛП. Нарощування розрядності АЛП у сучасних мікропроцесорах.

Будова АЛП

Арифметико-логічний пристрій (операційний пристрій) призначений для виконання арифметичних і логічних операцій над числами (словами), що надходять до нього. Основою АЛП є багаторозрядний суматор та набір регістрів для збереження операндів, результату обчислень та іншої інформації.

Пристрій керування призначений для організації процесу обчислень. Він координує дії АЛП, генеруючи у визначеній часовій послідовності керуючі сигнали, під дією яких у вузлах АЛП виконуються необхідні операції.

Сукупність АЛП і пристрою керування називають **процесорним пристроєм** або просто **процесором**. На рисунку 6.1 подано схему взаємодії основних компонентів процесора, включаючи АЛП.

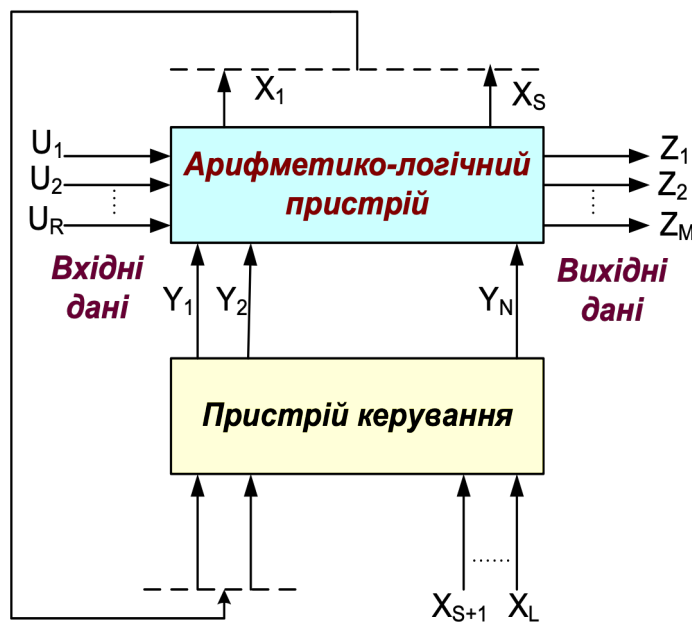


Рисунок 6.1 – Схема роботи процесорного пристрою

У цій схемі $y_1, \dots, y_N$ – керуючі сигнали для виконання визначених мікрокоманд, $x_1, \dots, x_S$ – сигнали станів цифрових автоматів АЛП, а $x_{S+1}, \dots, x_L$ – зовнішні сигнали.

Арифметико-логічний пристрій взаємодіє з іншими компонентами процесора через внутрішні шини даних і керуючих сигналів (рис. 6.2). Для виконання операцій він отримує операнди з регістрів загального призначення або з надоперативної пам'яті, виконує над ними задану дію – додавання,

віднімання, логічне «І», логічне «АБО», порівняння тощо і зберігає результат у відповідний регістр, або передає його назад у пам'ять. Керування роботою АЛП здійснює керуючий пристрій процесора, який формує необхідні керуючі сигнали залежно від коду виконуваної інструкції. Сам АЛП містить схеми для обробки переповнення при виконанні арифметичних операцій, формування ознак результату у випадку, наприклад, нульового результату, від'ємного числа, переповнення та ін. та установки цих відповідних прапорців у спеціальному регістрі стану.

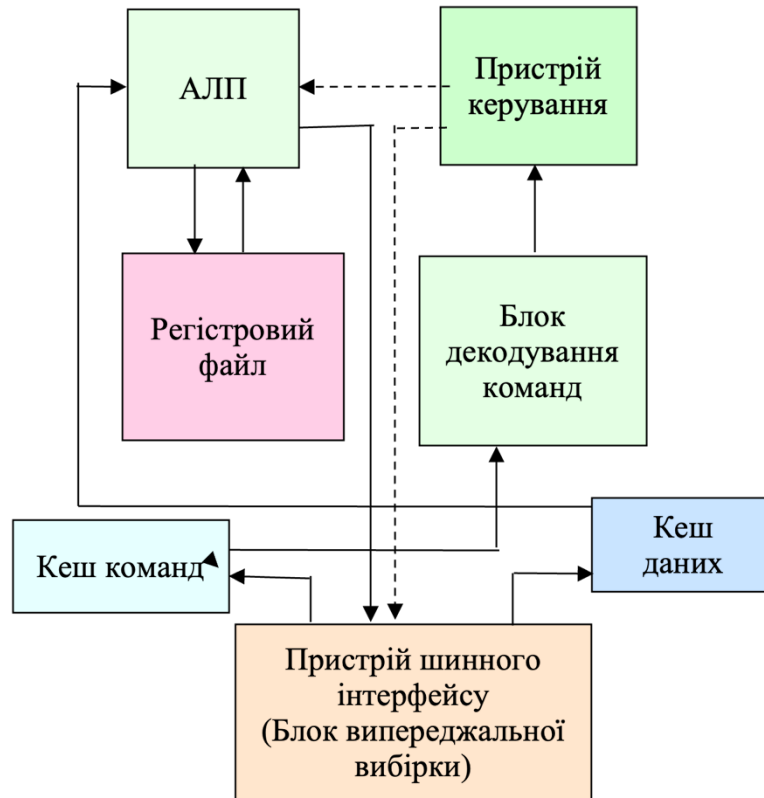


Рисунок 6.2 – Спрощена структура процесора

Крім основних арифметичних і логічних операцій, сучасні АЛП можуть також підтримувати операції з плаваючою комою, зсуви, обробку векторних даних та інші спеціалізовані функції, що прискорюють обчислення у складних задачах. Це дозволяє процесору ефективно виконувати широкий спектр інструкцій за короткий проміжок часу, підвищуючи загальну продуктивність системи.

Багато сучасних процесорів мають декілька АЛП, що дозволяє виконувати кілька операцій одночасно (паралельно), використовуючи принцип конвеєризації або суперскалярної архітектури. Такий підхід ще більше підвищує ефективність обробки даних і швидкість виконання програм.

Загальна схема роботи АЛП наступна. Вхідні дані подаються до АЛП через спеціальний інтерфейс. Це можуть бути два або більше операндів, що беруть участь у виконанні арифметичних або логічних операцій. Код операції надходить до АЛП зверху. Він визначає, яку саме операцію потрібно виконати над вхідними даними. Після обробки вихідні дані передаються далі – до регістрів або до інших частин процесора чи пам'яті. Сигнали станів, що

виходять униз від АЛП, містять інформацію про результат операції. Ці сигнали використовуються для прийняття рішень у подальших інструкціях процесора, наприклад, в умовних переходах (рис. 6.3).

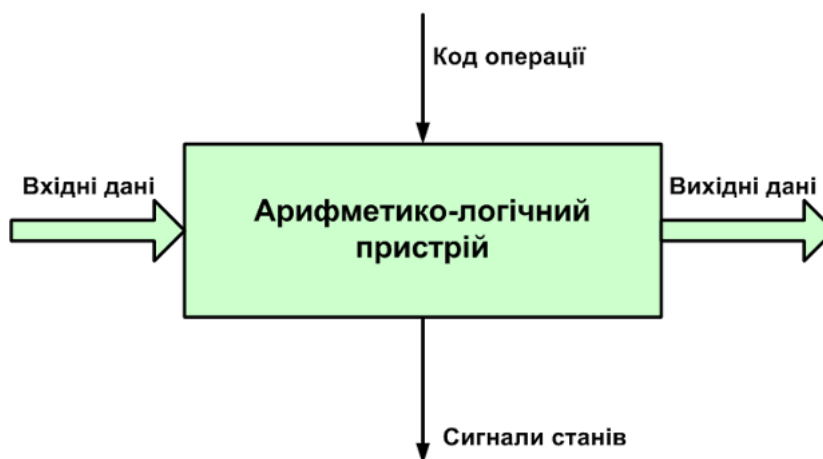


Рисунок 6.3 – Вхідні та вихідні сигнали АЛП

Таким чином, схема показує, що АЛП приймає вхідні дані та код операції, виконує відповідні обчислення і формує вихідні дані разом із сигналами стану, які описують властивості отриманого результату.

Арифметико-логічні пристрої класифікують за кількома ознаками.

За типом виконуваних операцій АЛП поділяються на арифметичні, логічні та комбіновані, які виконують як арифметичні, так і логічні дії.

За способом обробки даних виділяють паралельні АЛП, що обробляють кілька розрядів одночасно, та послідовні, які працюють із даними по одному розряду.

За типом числових форматів існують АЛП з фіксованою точкою та з плаваючою комою, що забезпечують різний рівень точності обчислень.

За організацією виконання операцій розрізняють конвеєрні АЛП, де обробка розбита на кілька стадій, і неконвеєрні, що виконують кожну операцію повністю перед обробкою наступної.

Також АЛП класифікують за спеціалізацією: універсальні, здатні виконувати широкий набір функцій, і спеціалізовані, оптимізовані для окремих задач, наприклад, криптографічних або мультимедійних обчислень.

Одноблокові та багатоблокові АЛП

Одноблокові арифметико-логічні пристрої являють собою такі пристрої, у яких усі функціональні елементи об'єднані в один загальний блок. В одному блоці реалізуються як арифметичні, так і логічні операції, а вибір необхідної дії визначається за допомогою кодів операцій і відповідних керуючих сигналів. Така організація є досить простою і компактною, але має обмежену продуктивність, оскільки одночасно виконується лише одна операція, що може створювати затримки при великій кількості обчислювальних задач.

Багатоблокові АЛП мають розділення на окремі функціональні блоки для різних типів операцій. Наприклад, один блок відповідає за арифметичні операції, інший – за логічні, ще інший може бути призначений для зсувів або

обробки чисел з плаваючою комою. Така структура дозволяє виконувати кілька операцій одночасно або швидше перемикатися між ними, що значно підвищує загальну продуктивність пристрою. Багатоблокові АЛП зазвичай використовуються в сучасних високопродуктивних процесорах, де важлива швидкість обробки великої кількості інструкцій.

Послідовні, послідовно-паралельні та паралельні АЛП

Залежно від способу обробки операндів АЛП діляться на послідовні, послідовно-паралельні та паралельні. В послідовних АЛП обробка операндів здійснюється послідовно в часі над кожним розрядом (рис. 6.4).

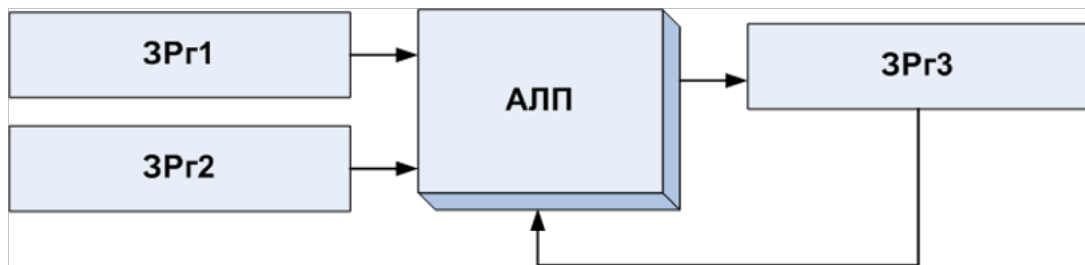


Рисунок 6.4 – Послідовний АЛП

Тут на вході АЛП є зсувні регістри ЗРг1 та ЗРг2, з яких дані порозрядно поступають на обробку. Результат з АЛП також порозрядно поступає в вихідний зсувний регістр ЗРг3.

У паралельному АЛП на вході АЛП є регістри Рг1 та Рг2, з яких дані паралельно поступають на обробку. Результат також паралельно поступає в вихідний регістр Рг3. Оскільки обробка здійснюється паралельно, вона виконується протягом лише одного такту незалежно від розрядності операндів. Тобто АЛП з паралельним способом обробки даних характеризується високою швидкістю, що і є причиною його широкого використання (рис. 6.5).

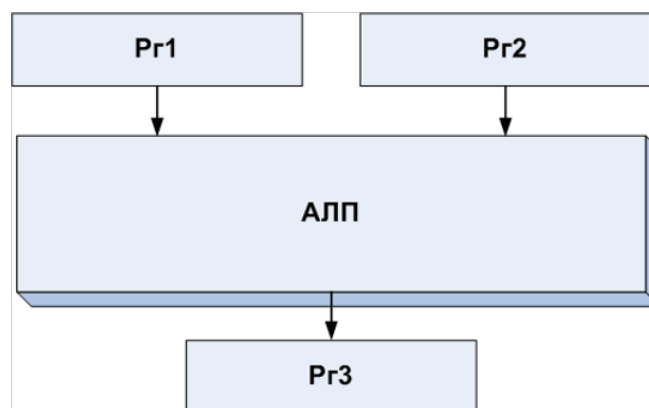


Рисунок 6.5 – Паралельний АЛП

Послідовно-паралельний спосіб обробки даних є проміжним стосовно швидкодії та затрат обладнання в порівнянні з вище розглянутими послідовним та паралельним способами (рис. 6.6).

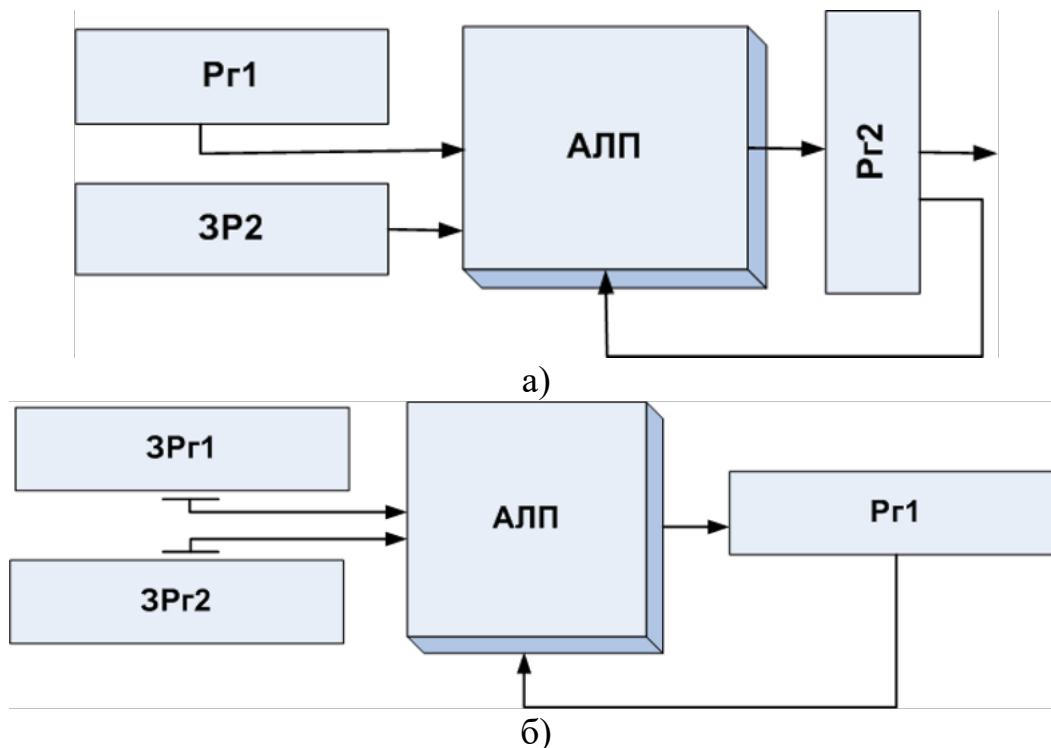


Рисунок 6.6 – Послідовно-паралельні АЛП:
а) з видачею проміжного результату; б) з групами вхідних розрядів

Тут одне з вхідних даних може поступати на обробку в АЛП паралельно, а інше послідовно з видачею проміжного результату в паралельному коді, як це показано на рис. а, або вхідні дані можуть поступати в АЛП групами по k і m розрядів, як це показано на рис. б, та подаватись в вихідний регістр паралельно, або також групами.

Нарощування розрядності АЛП у сучасних мікропроцесорах.

Нарощування розрядності арифметико-логічних пристроїв у сучасних мікропроцесорах є одним із основних способів підвищення їх обчислювальної потужності та ефективності. Розрядність АЛП визначає кількість бітів, які можуть бути оброблені одночасно за одну операцію. Історично спочатку використовувалися 4-, 8- та 16-розрядні АЛП, однак із розвитком технологій виникла потреба у збільшенні розрядності для обробки більших обсягів даних за короткий час. Внаслідок цього, з'явилися 32-розрядні, а потім і 64-розрядні мікропроцесори, де арифметико-логічні пристрої здатні працювати з більш довгими словами даних, що дозволяє збільшити точність обчислень та обсяг адресованої пам'яті.

Нарощування розрядності також важливе для виконання складних математичних обчислень, обробки мультимедійної інформації, роботи із великими базами даних та моделювання складних систем у реальному часі. У багатьох сучасних мікропроцесорах реалізовано ще ширшу розрядність у спеціалізованих модулях, наприклад, для обробки графіки або векторних обчислень (SIMD), де використовуються 128-, 256- або навіть 512-бітні блоки. Проте збільшення розрядності вимагає більш складної апаратної реалізації, підвищеного енергоспоживання та складнішого управління потоками даних,

тому проектування таких АЛП вимагає ретельного балансування між продуктивністю та енергоефективністю.

Контрольні питання:

1. Що таке арифметико-логічний пристрій (АЛП) і яку роль він виконує у процесорі?
2. Які основні функціональні елементи входять до складу АЛП?
3. Як працює багаторозрядний суматор у складі АЛП?
4. Для чого в АЛП використовуються регістри?
5. Що таке код операції в контексті роботи АЛП?
6. Які особливості будови одноблокових АЛП?
7. Які переваги та недоліки мають одноблокові АЛП?
8. У чому полягає принцип роботи багатоблокових АЛП?
9. Як багатоблокова структура впливає на продуктивність АЛП?
10. Чим відрізняється виконання операцій в одноблокових і багатоблокових АЛП?
11. Що таке послідовний АЛП і як він обробляє дані?
12. Як працює послідовно-паралельний АЛП?
13. Які переваги має паралельний АЛП порівняно з послідовним?
14. У яких випадках доцільно використовувати послідовний АЛП?
15. Як нарощування розрядності АЛП впливає на обчислювальну здатність мікропроцесора?
16. Які труднощі виникають при збільшенні розрядності АЛП?
17. Яка розрядність є типовою для сучасних мікропроцесорів загального призначення?
18. Для чого у сучасних процесорах використовуються 128-, 256- і 512-бітні блоки в АЛП?
19. Як збільшення розрядності АЛП впливає на енергоспоживання процесора?
20. Яким чином нарощування розрядності АЛП пов'язане з розвитком багатоядерних та векторних обчислень?

ТЕМА 7. ПРИСТРІЙ КЕРУВАННЯ У СКЛАДІ ПРОЦЕСОРА

Функції та методи побудови пристрою керування. Організація роботи пристрою мікропрограмного керування. Види мікропрограмування. Методи забезпечення координації ресурсів, передбачення переходів, розширення набору інструкцій (AVX, SSE для x86, SIMD для ARM).

Функції та методи побудови пристрою керування

Пристрій керування (ПК) виробляє послідовність сигналів, необхідних для виконання команди та програми в цілому. Команда виконується за один або декілька тактів, в кожному із яких виконується одна або декілька мікрооперацій (рис. 7.1). Мікрооперація – це деяка елементарна дія передачі або перетворення інформації, яка ініціюється поступленням керуючого сигналу (мікронаказу) на вхід керування відповідного пристрою. Для реалізації команди необхідно на відповідні керуючі входи подати розподілену в часі послідовність керуючих сигналів.

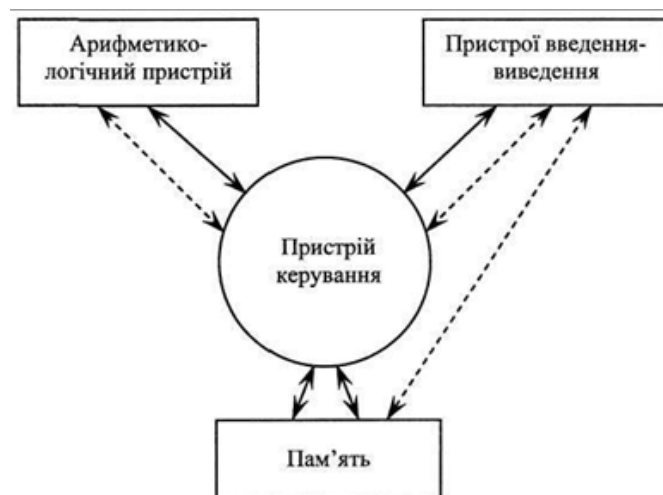


Рисунок 7.1 – Схема взаємодії пристрою керування з основними блоками CPU

Існують два основних методи побудови логіки формування керуючих сигналів:

Принцип схемної («жорсткої» або «запаяної») логіки: для кожної команди процесора існує набір логічних схем, які в потрібних тактах збуджують відповідні сигнали керування.

Принцип мікропрограмного керування передбачає формування керуючих сигналів за вмістом регістра мікрокоманд, в який мікрокоманди записуються із пам'яті мікрокоманд. Шляхом послідовного зчитування мікрокоманд із пам'яті в цей регістр організується потрібна послідовність керуючих сигналів.

Методи проектування ПК з жорсткою логікою, які застосовуються на практиці, часто є спеціально створеними для побудови конкретного ПК і

евристичними за природою, тому не можуть легко бути формалізованими. Найбільш розповсюдженими є методи:

- стандартний алгоритмічний підхід до проектування послідовнісних схем (метод таблиць станів);
- евристичні методи, що ґрунтуються або на використанні тактованих елементів часової затримки для побудови часової діаграми керуючих сигналів, або на використанні лічильників для побудови часової діаграми керуючих сигналів.

Найбільш формалізованим є метод таблиць станів, який передбачає розгляд ПК як цифрового автомату (ЦА), тобто логічного пристрою, який забезпечує формування сигналів керування за відповідним алгоритмом з врахуванням своїх внутрішніх станів.

ЦА можна подати у вигляді його математичної (абстрактної) і структурної моделей, які відповідно називаються **абстрактним та структурними автоматами**. Абстрактну модель використовують на першому етапі проектування, коли описують функціонування автомату, тобто правила переробки вхідної інформації у вихідну. На цьому етапі автомат подається у вигляді «чорної скриньки». Розгляд абстрактної моделі ЦА дозволяє проводити його попередню оптимізацію ще до етапу структурного синтезу. Структурну модель застосовують для побудови схеми ЦА.

Організація роботи пристрою мікропрограмного керування

Основні принципи, які покладені в основу організації роботи пристрою мікропрограмного керування (рис. 7.2):

1. Всі мікронакази (мікрооперації), які повинні бути виконані в одному такті роботи комп'ютера, збираються в одне керуюче слово, яке називають мікрокомандою (МК).
2. Кожній команді з системи команд комп'ютера ставиться у відповідність послідовність МК, необхідних для її виконання, тобто мікропрограма виконання команди в комп'ютері.
3. Всі МК зберігаються в пам'яті. Це може бути основна пам'ять комп'ютера, але в більшості комп'ютерів для зберігання МК використовується окрема пам'ять, яку називають пам'яттю МК.

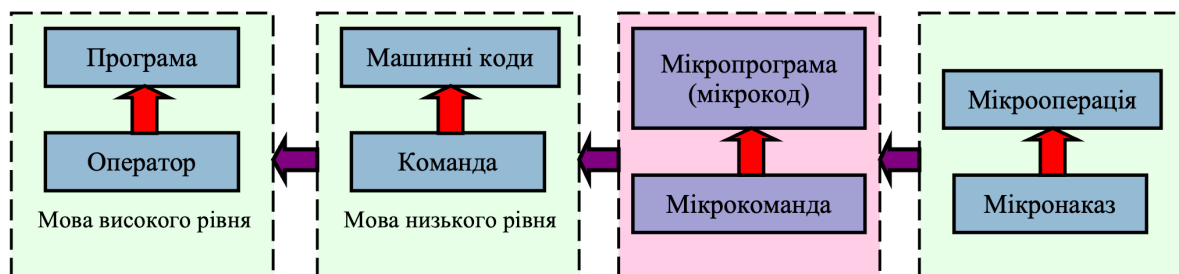


Рисунок 7.2 – Схема формування організації ПК

Для реалізації деякої команди необхідно зчитати з пам'яті МК відповідну послідовність МК (мікропрограму) та подати розподілену в часі послідовність керуючих сигналів на відповідні керуючі входи вузлів комп'ютера.

Є багато шляхів можливої організації мікропрограм в пам'яті МК. Один з основних – розміщення МК в пам'яті послідовно. Крім послідовності МК для кожної машинної команди, пам'ять МК містить МК для вибірки команд з основної пам'яті, запуску переривання, та деяких інших дій керування. Після того, як ПК вибирає машинну команду з основної пам'яті і розміщує її в РгК, він генерує адресу точки входу для КОП команди, яка є адресою першої МК мікропрограми.

Завдання вибору формату МК є досить складним. З одного боку, МК повинна вміщувати коди керування вузлами комп'ютера, забезпечувати організацію МК у мікропрограму та можливість зміни порядку МК у мікропрограмі. З іншого боку, необхідно мінімізувати кількість бітів в МК, кількість слів в пам'яті МК та час на виконання мікропрограми.

Види мікропрограмування

За способом формування керуючих сигналів розрізняють горизонтальне і вертикальне мікропрограмування.

При горизонтальному мікропрограмуванні (рис. 7.3) кожний розряд поля коду мікрооперації МК формує один керуючий сигнал для відповідного входу керування функціонального вузла комп'ютера.



Рисунок 7.3 – Схема формування МК при горизонтальному мікропрограмуванні

Кількість розрядів МК визначається з виразу

$$M=n+k+m,$$

де $n = \log_2 N$ – це розрядність адреси МК;

N – кількість МК в пам'яті;

k – кількість мікронаказів, необхідних для керування вузлами ПК;

m – кількість мікронаказів, необхідних для керування вузлами комп'ютера.

В цьому випадку непотрібний дешифратор МК; але, оскільки у комп'ютерах кількість мікронаказів може досягати декількох сотень, МК має дуже багато розрядів.

У вертикальному мікропрограмуванні, навпаки, поле коду мікрооперації складається з компактних кодувань, де кожен код відповідає певній групі керуючих сигналів або певній мікрооперації. Це дозволяє зменшити довжину мікрокоманд і спростити їхнє зберігання в мікропрограмній пам'яті, але

знижує швидкодію, оскільки одночасно може виконуватися менша кількість мікрооперацій. Вертикальне мікропрограмування частіше застосовується там, де критичне значення має компактність коду і економія ресурсів пам'яті.

Методи забезпечення координації ресурсів, передбачення переходів, розширення набору інструкцій (AVX, SSE для x86, SIMD для ARM).

У сучасних процесорах для досягнення ефективної координації ресурсів застосовуються різноманітні методи. Один із основних способів – це перевизначення ресурсів, що дозволяє уникати блокувань через залежності між інструкціями, які використовують однакові ресурси. Це особливо важливо в суперскалярних архітектурах, де процесор може виконувати кілька інструкцій одночасно. Завдяки перевизначенню ресурсу можна запобігти «штучним» залежностям між інструкціями, що дозволяє значно збільшити продуктивність процесора.

Передбачення переходів є критично важливим для зниження затримок у процесорах, оскільки переходи (if-else конструкції, цикли, виклики функцій) можуть призводити до значних затримок, якщо процесор не здогадується про напрямок переходу. Використовуються різні методи передбачення, такі як статичне передбачення (на основі попереднього досвіду, напр., всі переходи приймаються як "вірні") та динамічне передбачення (з використанням спеціальних буферів і алгоритмів, наприклад, буферів історії переходів). Такі технології використовуються у сучасних процесорах, щоб мінімізувати час простою в очікуванні результатів попередніх інструкцій.

Розширення набору інструкцій дозволяють процесорам виконувати більш складні операції за допомогою нових інструкцій, що працюють з кількома даними одночасно. Це особливо важливо для обчислень, які потребують високої паралельності, таких як обробка відео, графіки, машинне навчання та наукові розрахунки. AVX (Advanced Vector Extensions) і SSE (Streaming SIMD Extensions) для x86 архітектури дозволяють обробляти векторні дані, а SIMD (Single Instruction, Multiple Data) для ARM дає змогу обробляти декілька елементів одночасно в одному такті процесора, що значно збільшує швидкість виконання таких операцій.

AVX – це набір розширень для процесорів архітектури x86, який дозволяє виконувати більш складні векторні операції, що мають високу паралельність, та є частиною архітектури Intel. AVX розширює можливості SIMD (Single Instruction, Multiple Data), дозволяючи одночасно обробляти векторні дані більшої довжини порівняно з попередніми розширеннями, такими як SSE. Однією з основних особливостей AVX є використання 256-бітних векторних регістрів, що дозволяє обробляти більші обсяги даних в одному такті процесора.

Початкові версії AVX називались AVX1 і були введені з архітектурою Intel Sandy Bridge. Вони дозволяли виконувати 8 одночасних 32-бітних операцій з плаваючою комою або 4 одночасні 64-бітні операції. Вдосконалення в AVX2 додавало можливості для більш ефективної обробки цілих чисел, а AVX-512, що з'явилося з Intel Skylake-X, дозволяло

використовувати 512-бітні векторні регістри та ще більше підвищує продуктивність у багатозадачних та паралельних обчисленнях.

SE – це ще один набір розширень для процесорів архітектури x86, також спрямований на паралельне оброблення даних. SSE був вперше введений Intel з архітектурою Pentium III і включає в себе SIMD інструкції, що дозволяють обробляти кілька елементів за один такт процесора. SSE використовує 128-бітні регістри для обробки даних, що дає змогу одночасно обробляти чотири 32-бітні значення з плаваючою комою або два 64-бітні значення.

З часом SSE було розширено з додаванням нових версій, таких як SSE2, SSE3, SSE4.1, SSE4.2, і так далі. Кожне з розширень дозволяло зменшити затримки та підвищити ефективність обробки конкретних типів даних. Наприклад, SSE4.2 додало нові інструкції для швидкого пошуку підстрічок у рядках і обчислення хешів, стали корисними для алгоритмів пошуку в базах даних та криптографії.

Основні відмінності між AVX і SSE:

1. Розмір векторів. SSE використовує 128-бітні регістри, а AVX – 256-бітні (AVX2). Це дозволяє AVX обробляти більше даних за один такт.

2. Продуктивність: AVX має більшу пропускну здатність завдяки більш довгим векторним регістрами, що дає змогу обробляти більші набори даних одночасно. Це дає значне підвищення продуктивності для програм, які використовують паралельну обробку даних (наприклад, графіка, наукові обчислення, машинне навчання).

3. Сумісність: SSE має широку сумісність із старими процесорами x86, тоді як AVX з'явився в новіших процесорах і потребує спеціальних розширених інструкцій.

Обидва розширення, і AVX, і SSE, активно використовуються в сучасних архітектурах для високопродуктивних обчислень, обробки медіа та графіки, а також в наукових та інженерних задачах.

Контрольні питання:

1. Що таке пристрій керування в комп'ютерних системах і яку роль він відіграє у роботі процесора?

2. Як організована робота пристрою мікропрограмного керування? Охарактеризуйте основні етапи виконання мікропрограм.

3. Які основні види мікропрограмування існують, і в чому їх відмінності?

4. Що таке мікропрограма і як вона реалізується в пристрої мікропрограмного керування?

5. Як забезпечується координація ресурсів в процесорах? Які методи використовуються для запобігання конфліктам між інструкціями?

6. Яким чином передбачення переходів впливає на продуктивність процесора? Охарактеризуйте основні методи передбачення переходів.

7. Що таке розширення набору інструкцій AVX і SSE для архітектури x86? Як вони підвищують ефективність обробки векторних даних?

ТЕМА 8. КОДУВАННЯ КОМАНД В КОМП'ЮТЕРІ

Структура команди. Вибірка команд, розпізнавання типів виконуваних операцій. Звернення до пам'яті, порядок виконання та цикли виконання команд. Типи архітектур системи команд комп'ютера.

Структура комп'ютерної команди

Загальна структура команди комп'ютера наведена на рисунку 8.1:

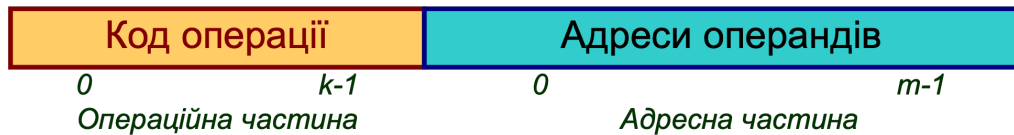


Рисунок 8.1 – Схема команди комп'ютера

Операційна частина містить код операції (КОп), який визначає: тип операції (запис, читання, додавання, множення, пересилання інформації та інші); тип операндів; формат операндів; довжину коду команди; спосіб адресації операндів та інколи іншу інформацію.

Адресна (операндна) частина команди містить інформацію про адреси операндів (або самі операнди) і результату операції, а в деяких випадках, інформацію про адресу наступної команди.

Поле коду операції (КОП) займає k розрядів. Ним може бути закодовано до $N = 2^k$ різних операцій. Кількість двійкових розрядів, які відводяться під код операції, вибирається таким чином, щоб ними можна було закодувати всі виконувані в даному комп'ютері операції. Якщо деякий комп'ютер може виконувати N_c різних операцій, то мінімальна розрядність поля коду операції k визначається наступним чином: $k = \lceil \log_2 N_c \rceil$, де вираз в дужках означає заокруглення до більшого цілого.

Поле адреси (адресна частина) займає m розрядів. В ньому знаходяться адреси операндів. Кожна адреса займає m_i розрядів, де i – номер адреси ($i=1,2,\dots,l$), l – кількість адресних полів. Кожною адресою можна адресувати пам'ять ємністю 2^{m_i} слів. Детальна інформація про зв'язок між ємністю пам'яті та розрядністю адреси наведена в розділі 9, в якому описана будова пам'яті. СА- спосіб адресації операндів.

Розмір команди $k + m$ повинен бути узгодженим з розміром даних, тобто бути з ним однаковим або кратним цілому числу, що спрощує організацію роботи з пам'яттю. Як правило, розмір команди може бути 8, 16, 32 чи 64 біти.

Вибірка команд, розпізнавання типів виконуваних операцій

Вибірка команд – це перший етап у циклі виконання інструкцій процесора. На цьому етапі процесор отримує з оперативної пам'яті (або кеш-пам'яті) код команди, адресу якої вказує лічильник команд. Після вибірки команда передається в блок дешифрації для подальшої обробки, а лічильник команд оновлюється, щоб вказувати на наступну інструкцію. Швидкість та

ефективність вибірки команд є критичними для загальної продуктивності процесора, тому сучасні процесори використовують кеш команд і механізми попередньої вибірки. Після вибірки команда потрапляє в блок дешифрації, де відбувається розпізнавання її типу. Процесор аналізує закодовану інформацію в інструкції, визначає, який саме тип операції необхідно виконати: арифметична, логічна, передавання даних, керування переходами тощо, які операнди будуть задіяні, і яку функціональну одиницю потрібно активувати, наприклад, ALU, блок доступу до пам'яті або блок управління переходами. У сучасних процесорах дешифрація може здійснюватися паралельно для кількох інструкцій одразу, що дозволяє пришвидшити їх обробку.

Звернення до пам'яті, порядок виконання команд та цикли виконання команд

Звернення до пам'яті – це процес читання або запису даних у пам'ять комп'ютера. Під час виконання команди процесор може звертатися до пам'яті для отримання операндів, або для збереження результатів обчислень. Адреса пам'яті для звернення може бути задана явно в команді або обчислюватися в процесі виконання. Звернення до пам'яті включає два основні етапи: обчислення фізичної адреси та безпосереднє читання/запис даних. Сучасні процесори використовують кеш-пам'ять для пришвидшення доступу, а також можуть виконувати кілька звернень паралельно для підвищення продуктивності. Процес виконання відбувається у циклі.

Цикл команд (іноді називається циклом вибірка-декодування-виконання) є основним операційним процесом комп'ютера. Це процес, за допомогою якого комп'ютер отримує команду програми з пам'яті, визначає, які дії виконуються командою, і виконує ці дії (рис. 8.2).

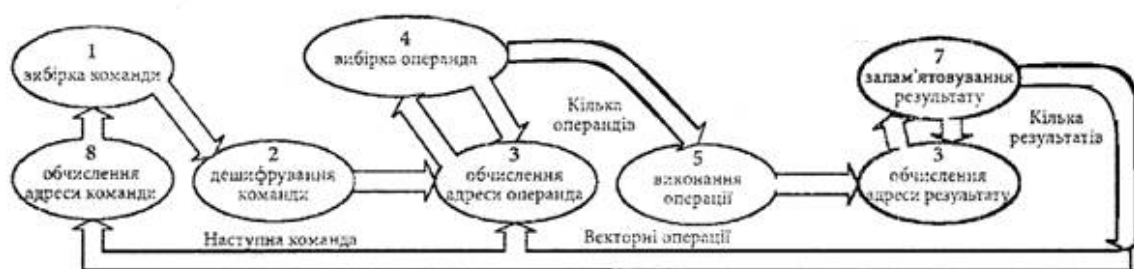


Рисунок 8.2 – Схема циклу операційного процесу комп'ютера [11]

Наведена схема описує етапи обробки інструкції у процесорі. Це класичний цикл роботи пристрою керування. Спочатку відбувається вибірка команди: процесор вибирає команду з пам'яті за допомогою адреси, яку вказує лічильник команд. Вибрана команда надходить у блок дешифрування. Наступний крок – дешифрування команди. Тут команда аналізується: визначається її тип, кількість операндів та формат. Далі відбувається обчислення адреси операндів. Якщо команда потребує операндів із пам'яті, обчислюється їхня адреса на основі інформації, отриманої після дешифрування.

Відповідно до обчислених адрес здійснюється вибірка необхідних даних із пам'яті або регістрів для подальшого використання і виконання операції – операнди передаються в арифметико-логічний пристрій, або інший функціональний блок, для виконання вказаної в команді операції. Якщо результат потрібно зберегти у пам'яті або у спеціальний регістр, обчислюється адреса для його запису. Далі йде запам'ятовування результату. Отриманий результат записується у визначене місце: регістр або оперативну пам'ять і слідом відбувається обчислення адреси наступної команди. Під час цього виконання паралельно готується адреса для вибірки наступної інструкції, зазвичай шляхом інкрементації лічильника команд або шляхом обчислення нової адреси у випадку переходів.

Після кожного етапу дані переходять до наступного блоку, а після завершення виконання команди управління повертається на вибірку наступної команди, проте, частина етапів (наприклад, обчислення адреси операндів і результатів) може працювати паралельно.

Типи архітектур системи команд комп'ютера

Класифікація архітектури комп'ютера за типом адресованої пам'яті також враховує, де саме зберігаються проміжні результати обчислень і як вони обробляються. Одним із типів є акумуляторна архітектура (рис. 8.3). В такій архітектурі основним елементом для збереження проміжних результатів виступає акумулятор – спеціальний регістр. Операції виконуються над даними, що зчитуються з пам'яті в акумулятор, обробляються там і результати записуються назад у пам'ять. Цей підхід є простим в реалізації і вимагає мінімальної кількості операндів в інструкціях, однак часто призводить до великої кількості звернень до пам'яті, що знижує продуктивність.



Рисунок 8.3 – Акумуляторна архітектура [11]

Інший тип – це стекова архітектура (рис. 8.4) , в якій проміжні результати обчислень зберігаються в стеку – спеціальній області пам'яті, що працює за принципом «останній увійшов – перший вийшов» – LIFO. Процесор автоматично працює з верхівкою стеку, тому інструкції стають коротшими, бо немає потреби явно вказувати операнди. Стекова архітектура зменшує складність управління операндами, але ускладнює оптимізацію виконання команд, оскільки обмежена прямим доступом тільки до верхніх елементів стеку.

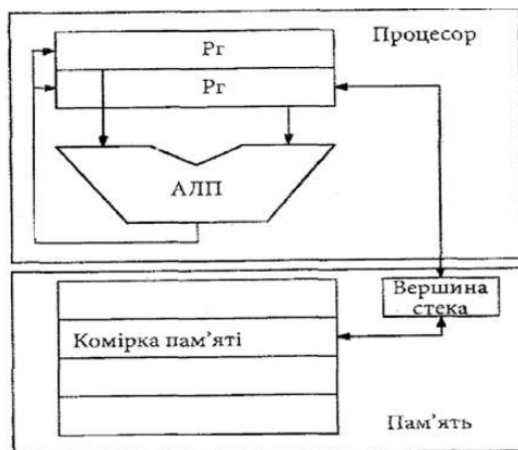


Рисунок 8.4 – Стекова архітектура [11]

Найпоширенішою в сучасних процесорах є архітектура на основі регістрів загального призначення. У цій архітектурі проміжні результати обчислень зберігаються у великій кількості регістрів всередині процесора. Інструкції вказують явно, з якими регістрами працювати, що дає велику гнучкість і дозволяє ефективно оптимізувати виконання програм.

В архітектурі типу пам'ять–пам'ять (рис. 8.5) усі операції виконуються безпосередньо над даними, які зберігаються в пам'яті.

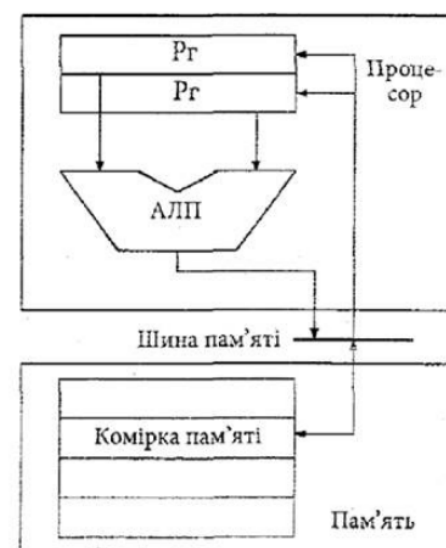


Рисунок 8.5 – Архітектура на основі РЗП типу пам'ять-пам'ять [11]

Тут операнди для обчислень зчитуються прямо з пам'яті, а результати також безпосередньо записуються назад у пам'ять. Така схема мінімізує використання регістрів процесора, але водночас спричиняє велику кількість звернень до оперативної пам'яті, що знижує продуктивність через повільніший доступ до даних порівняно з регістрами.

В архітектурі типу регістр–пам'ять (рис. 8.6) одна з операндних величин може знаходитися в регістрі, а інша – у пам'яті. Результат операції може записуватися або в регістр, або безпосередньо в пам'ять. Така модель дає гнучкість: процесор може працювати як із швидкою пам'яттю регістрів, так і безпосередньо з основною пам'яттю, зменшуючи кількість проміжних копіювань даних. Водночас вона вимагає більш складного управління доступом до пам'яті на рівні апаратури.



Рисунок 8.6 – Архітектура на основі РЗП типу регістр-пам'ять [11]

Нарешті, в архітектурі типу регістр–регістр усі обчислення виконуються тільки над даними, що зберігаються в регістрах. Перед виконанням операцій необхідно завантажити операнди з пам'яті в регістри, а після обчислень результати за потреби записуються назад у пам'ять. Такий підхід мінімізує кількість звернень до пам'яті під час обчислень і дозволяє максимально використовувати швидкодію регістрів, тому він характерний для більшості сучасних RISC-процесорів.

Дані для обробки в архітектурі типу регістр–регістр завантажуються із основної пам'яті в регістри за допомогою окремих команд завантаження. Спочатку процесор виконує спеціальні інструкції, які зчитують необхідні операнди із пам'яті в певні регістри загального призначення. Після цього арифметичні або логічні операції виконуються виключно між регістрами без подальших звернень до зовнішньої пам'яті. Результати обчислень залишаються в одному з регістрів і можуть зберігатися там для подальшого використання в наступних операціях або бути записані назад у пам'ять за допомогою окремих інструкцій збереження.

Такий принцип «чистого» розділення завантаження/збереження даних і виконання обчислень характерний для архітектур із філософією RISC (Reduced Instruction Set Computer). Він дозволяє спростити конвеєризацію інструкцій, оптимізувати апаратну реалізацію процесора й забезпечити високу частоту виконання команд. Обробка даних у регістрах значно швидша, ніж у пам'яті, оскільки регістри мають мінімальну затримку доступу, а також їх адресація відбувається простіше й швидше. Завдяки цьому архітектура типу регістр–регістр (рис. 8.7) забезпечує ефективне використання апаратних ресурсів і є основою більшості сучасних високопродуктивних процесорів.



Рисунок 8.7 – Архітектура на основі РЗП типу регістр-регістр [11]

Завдяки використанню регістрів кількість звернень до зовнішньої пам'яті значно зменшується, що суттєво підвищує швидкість обробки даних. Цей тип архітектури є основою для більшості сучасних мікропроцесорів.

Контрольні питання:

1. Що таке структура команди в архітектурі комп'ютера?
2. Які основні компоненти складають команду в комп'ютерній архітектурі?
3. Як відбувається вибірка команд у процесорі?
4. Які типи операцій можуть бути виконані в межах однієї команди?
5. Як процесор розпізнає тип операції, що виконується командою?
6. У чому полягає роль коду операції (opcode) в команді процесора?
7. Які типи операцій можуть бути реалізовані в архітектурі процесора (арифметичні, логічні, передача даних тощо)?
8. Що таке звернення до пам'яті, і які команди включають таке звернення?
9. Як визначається порядок виконання команд в процесорі?
10. Що таке цикл виконання команди, і які етапи він включає?
11. Як організоване звернення до пам'яті в процесі виконання команд?

12. Що таке пряме та непряме звернення до пам'яті?
13. Як цикли виконання команд впливають на ефективність обробки команд процесором?
14. Які основні типи архітектур системи команд існують (CISC, RISC, VLIW)?
15. Які переваги та недоліки архітектур CISC і RISC?
16. Яким чином різні типи архітектур системи команд впливають на загальну продуктивність комп'ютерних систем?

ТЕМА 9.

СИСТЕМИ КОМАНД КОМП'ЮТЕРА. АРХІТЕКТУРИ CISC ТА RISC

Вимоги до процесора комп'ютера з простою та комплексною системами команд. Основні відмінності архітектур CISC та RISC. Приклади сучасних архітектур системи команд на базі процесорів Intel Core i9, AMD Ryzen 9 та ARM Cortex, Apple M4.

Вимоги до процесора комп'ютера з простою та комплексною системами команд

Одна з основних задач розвитку архітектури процесорів – усунення проблеми семантичного розриву між складними операторами мов програмування високого рівня та достатньо простими машинними операціями, що виконуються в процесорі, при постійному рості вимог до продуктивності вимагає пошуку шляхів зменшення семантичного розриву між високорівневими мовами програмування і низькорівневими машинними операціями, виконуваними процесором. Це завдання стає дедалі важливішим через постійне зростання вимог до продуктивності та ефективності комп'ютерних систем.

Одним із шляхів усунення семантичного розриву є використання складних команд, які можуть виконувати кілька операцій за один цикл. Архітектури, як CISC (Complex Instruction Set Computing), пропонують такі команди, що виконують високорівневі операції (наприклад, множення, ділення або обробку рядків) без необхідності виконувати кілька простих інструкцій. Це дозволяє зменшити кількість команд, що необхідно виконати для реалізації певної задачі, наближаючи систему до семантики мов програмування.

За допомогою спеціалізованих інтерпретаторів та компіляторів, які здатні перетворювати високорівневі конструкції в оптимізовані низькорівневі інструкції, можна значно зменшити розрив між рівнями. Більш ефективні алгоритми для оптимізації коду дозволяють компіляторам генерувати код, що найкраще використовує можливості апаратного забезпечення, і це може включати, наприклад, розпаралелювання обчислень або злиття кількох операцій. Ще одним напрямком є використання адаптивних та інтелектуальних архітектур, які здатні адаптуватися до різних типів програм. Наприклад, деякі сучасні процесори можуть автоматично налаштовувати свої параметри виконання залежно від характеру виконуваної програми, що дозволяє значно знизити рівень семантичного розриву.

Одним із найпродуктивніших способів спрощення команд і зменшення складності декодування стала підтримка високорівневих концепцій у апаратному забезпеченні. З розвитком процесорів з **архітектурою RISC (Reduced Instruction Set Computing)** було досягнуто значних успіхів. Це дозволяє зменшити витрати часу на виконання команд і оптимізувати виконання програм. Однак для забезпечення максимальної ефективності програмування, потрібно було б підтримати високорівневі абстракції, такі як

керування потоками або спеціалізовані операції (наприклад, робота з великими обсягами даних).

Зачатки архітектури RISC мають своє коріння в комп'ютерах CDC6600, розроблених Джином Торнтоном, Сеймуром Креєм та іншими інженерами, які вперше усвідомили важливість спрощення набору команд для створення швидких обчислювальних машин. Вони зрозуміли, що зменшення складності інструкцій дозволить значно підвищити продуктивність процесора, оскільки зменшиться час на їх виконання і декодування. Ця ідея згодом стала основою для подальших розробок у галузі комп'ютерної архітектури, включаючи серії суперкомп'ютерів компанії Cray Research, де Крей успішно застосував принципи спрощення архітектури для досягнення високої продуктивності.

Остаточне формулювання концепції RISC в сучасному її розумінні стало можливим завдяки трьом дослідницьким проектам. Процесор 801 компанії IBM, процесор RISC, розроблений в університеті Берклі, і процесор MIPS, створений в Стенфордському університеті, стали основними етапами на шляху розвитку цієї архітектури. Кожен з цих проектів зробив важливий внесок у розвиток ідей, які лягли в основу RISC, акцентуючи увагу на спрощенні команд, зменшенні циклів виконання та ефективності використання регістрів. Ці досягнення дозволили сформулювати концепцію RISC, яка стала стандартом для створення процесорів з високою продуктивністю та оптимізованою роботою з пам'яттю.

Основна концепція архітектури RISC полягає в спрощенні набору інструкцій процесора з метою підвищення швидкості виконання програм (рис. 9.1).

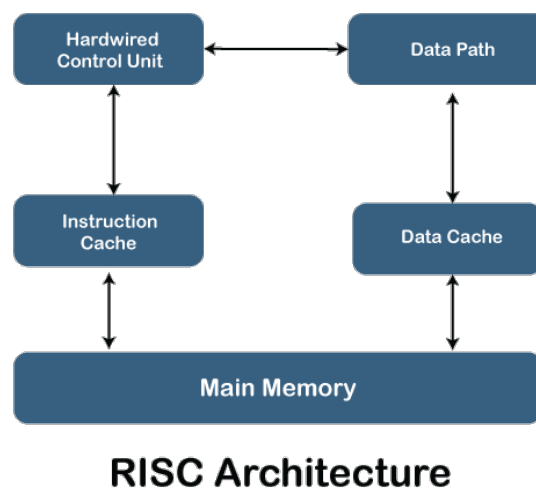


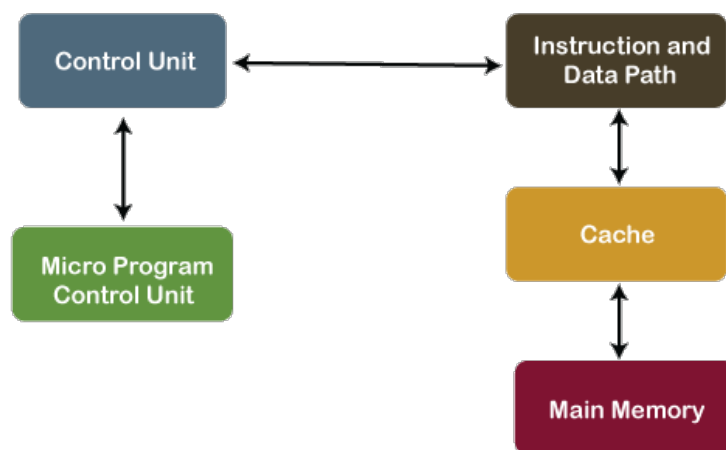
Рисунок 9.1 – Схема архітектури RISC [12]

Це досягається через кілька ключових принципів. По перше, у RISC процесорах кожна інструкція виконує лише одну просту операцію, зазвичай за один або кілька тактів процесора. Це контрастує з **архітектурами CISC (Complex Instruction Set Computing)**, де кожна інструкція може виконувати кілька складних операцій одночасно.

У процесорах RISC велика частина обробки даних здійснюється за допомогою реєстрів, а не безпосереднього звернення до пам'яті. Це дозволяє значно зменшити час доступу до даних, оскільки реєстри набагато швидші за оперативну пам'ять. Архітектура RISC обмежується невеликою кількістю інструкцій, що дозволяє простіше та швидше їх декодувати та виконувати. Це робить процесор ефективнішим при виконанні базових операцій. У RISC-архітектурі операції доступу до пам'яті відокремлені від обчислювальних операцій. Тобто, для операцій з пам'яттю використовуються окремі інструкції, а всі обчислення проводяться між реєстрами. Це дозволяє спростити виконання програм і зменшити навантаження на процесор. Завдяки спрощеній структурі інструкцій, RISC процесори здатні ефективно використовувати пайплайн, при якому різні етапи обробки інструкції (вибірка, декодування, виконання) можуть виконуватись одночасно для кількох інструкцій. Це дозволяє значно підвищити продуктивність.

Таким чином, концепція RISC орієнтована на максимальну швидкість виконання простих інструкцій при мінімальних витратах на декодування та виконання, що забезпечує високу ефективність процесорів.

CISC (Complex Instruction Set Computing) – це архітектура комп'ютерних процесорів, яка характеризується наявністю великої кількості складних інструкцій, здатних виконувати багато операцій за один такт (рис. 9.2).



CISC Architecture

Рисунок 9.2 – Схема архітектури CISC [12]

Вона була розроблена для спрощення програмування, оскільки кожна інструкція може виконувати кілька завдань одночасно, таких як арифметичні операції, доступ до пам'яті, зміни в реєстрах тощо. У процесорах CISC інструкції можуть бути довгі та складні, що дозволяє зменшити обсяг коду, який потрібно написати для виконання певної задачі.

Однією з головних переваг CISC архітектури є те, що програми, написані на високорівневих мовах програмування, можуть бути безпосередньо перекладені в машинний код, що включає складні інструкції. Це дозволяє зменшити кількість інструкцій, необхідних для виконання завдань, що знижує

обсяг коду та робить його більш компактним. Наприклад, у процесорах CISC можна реалізувати команди, що поєднують операції з пам'яттю та арифметичні операції, що дозволяє зменшити потребу в численних зверненнях до пам'яті.

Проте архітектура CISC має і свої недоліки. Через складність інструкцій процесор потребує більше часу на їх декодування та виконання, що може знижувати загальну продуктивність. Крім того, складність командного набору збільшує складність розробки процесора і компіляторів. Однак, завдяки потужним оптимізаціям і кешуванням, процесори CISC залишаються ефективними для деяких типів завдань, де велика кількість складних операцій є необхідною, наприклад, у старих комп'ютерах або в специфічних застосунках, що вимагають великої обчислювальної потужності при роботі з великими обсягами даних.

Основні відмінності архітектур CISC та RISC

Різниця між архітектурами CISC та RISC полягає, перш за все, у складності набору команд та способі їх виконання. У CISC процесорах набір інструкцій є великим і різноманітним, включаючи складні команди, що виконують кілька операцій за один такт. Це дозволяє зменшити розмір програми, оскільки одна інструкція може виконати кілька дій, що є корисним для програмістів, але з іншого боку, процесор потребує більше часу на декодування і виконання таких інструкцій. Крім того, у CISC архітектурах доступ до пам'яті часто поєднаний із арифметичними операціями, що ще більше ускладнює виконання команд.

На відміну від цього, у RISC архітектурах набір команд є значно простішим і складається з основних інструкцій, кожна з яких виконує одну операцію за один або кілька тактів процесора. Це забезпечує значно швидше виконання команд, оскільки інструкції легші для декодування і виконання, і більшість операцій здійснюється за допомогою регістрів, що є значно швидшими за основну пам'ять. Однак, у разі використання RISC, для виконання складних операцій програмістам потрібно писати більше інструкцій, що збільшує розмір програми, але при цьому забезпечує вищу ефективність в умовах частих операцій з регістрами та менших циклів виконання (рис. 9.3).

Характеристика	CISC	RISC
Довжина команд	Варіюється	Однакова
Розташування полів в команді	Варіюється	Незмінне
Кількість регістрів	Декілька (часто спеціалізованих)	Багато регістрів загального призначення (переважно рівноправних)
Доступ до пам'яті	Може виконуватися як частина команд різних типів	Виконується тільки спеціальними командами

Рисунок 9.3 – Порівняльні характеристики архітектур CISC та RISC

Таким чином, основна різниця між CISC і RISC полягає в підході до складності команд. CISC фокусується на зменшенні кількості інструкцій, що потрібно написати для реалізації задачі, але за рахунок складнішого виконання кожної інструкції, що може знижувати загальну швидкість. В свою чергу, RISC орієнтований на спрощення команд, що дозволяє збільшити швидкість виконання, але вимагає більшого обсягу програмного коду для досягнення тих самих результатів.

Приклади сучасних архітектур системи команд на базі процесорів Intel Core i9, AMD Ryzen 9 та ARM Cortex, Apple M4

Сучасні процесори використовують різні архітектури системи команд (ISA), кожна з яких має свої переваги та особливості. Вибір між архітектурами CISC і RISC залежить від специфіки завдань, для яких призначена система, а також від оптимізації параметрів, таких як продуктивність, енергоспоживання та теплоємність. Серед найбільш популярних сучасних архітектур є процесори на базі Intel Core i9, AMD Ryzen 9, ARM Cortex та Apple M4. Кожна з цих архітектур має унікальні особливості, що дозволяють їй ефективно виконувати різноманітні обчислювальні задачі, від високопродуктивних обчислень до мобільних додатків та енергоефективних рішень. Розглянемо їх детальніше, щоб зрозуміти, які принципи і технології лежать в основі кожної архітектури.

Процесори Intel Core i9 (рис. 9.4) використовують архітектуру x86-64, яка є розвитком класичної архітектури CISC.

Технічні характеристики Intel Core i9-14900K	
Літографія	Intel 7
Кількість ядер	24
Кількість потоків	32
Р-ядер	8
Е-ядер	16
Максимальна частота	6 ГГц
Частота Р-ядер	5,8 ГГц
Частота Е-ядер	4,4 ГГц
Кеш третього рівня	36 MB Intel® Smart Cache
Базовий TDP	125 Вт
Turbo TDP	253 Вт
Тип ОЗП	DDR4/DDR5
Максимум ОЗП	192 ГБ
Вбудована графіка	Intel UHD 770
Підтримка інструкцій	Intel® SSE4.1, Intel® SSE4.2, Intel® AVX2

Рисунок 9.4 – Таблиця характеристик Intel Core i9-14900K [14]

Це означає, що процесор має великий набір складних інструкцій, здатних виконувати кілька операцій за один такт.

Завдяки цьому досягається висока гнучкість у виконанні різноманітних програм, особливо для завдань, які потребують значної обчислювальної потужності, таких як робота з мультимедійними даними чи інтенсивні обчислення. Проте, складність декодування та виконання інструкцій може знижувати ефективність у деяких сценаріях порівняно з архітектурами на основі RISC.

Процесори AMD Ryzen 9 базуються на архітектурі x86-64, подібно до Intel Core i9, але вони використовують технологію, розроблену AMD, під назвою "Zen". Архітектура Zen поєднує елементи як CISC, так і деякі принципи RISC для оптимізації продуктивності. Вона включає в себе кілька ядер, здатних працювати паралельно, і використовує технології, такі як симультанний багатозадачний обчислювальний потік, що забезпечує високу продуктивність у багатозадачних середовищах. Крім того, підтримка високих тактових частот і значних обсягів кеш-пам'яті дозволяє забезпечити ефективне виконання складних програм.

Архітектура ARM Cortex, яка лежить в основі багатьох мобільних пристроїв, таких як смартфони, планшети та деякі ноутбуки, базується на принципах RISC. Вона має набір інструкцій, орієнтований на простоту і швидкість виконання, що дозволяє досягти високої енергоефективності при збереженні хороших показників продуктивності. ARM Cortex використовує менш складні інструкції порівняно з x86-64, що дозволяє зменшити споживану потужність і знизити тепловиділення, що є критичним для мобільних пристроїв.

Процесор Apple M4 є частиною лінійки чіпів Apple Silicon, що також базуються на архітектурі ARM. Архітектура ARM в чіпах Apple розширена і модифікована для оптимізації роботи з операційною системою macOS і специфічними завданнями, такими як машинне навчання та графічні обчислення. Чіпи Apple M4 мають набір інструкцій, який поєднує ефективність RISC з додатковими елементами, оптимізованими для специфічних завдань, таких як нейронні мережі та графіка. Завдяки цьому, процесори M4 досягають високої продуктивності при низькому енергоспоживанні, що робить їх ідеальними для інтеграції в продуктах Apple, таких як ноутбуки та планшети.

Контрольні питання:

1. Які основні типи систем команд існують?
2. Які основні вимоги до процесора комп'ютера з простою системою команд?
3. Які вимоги ставляться до процесора з комплексною системою команд?
4. Які основні відмінності між архітектурами CISC та RISC?
5. Як архітектура CISC впливає на продуктивність та складність декодування інструкцій?

6. Які переваги архітектури RISC у порівнянні з CISC?
7. Як використання регістрів в архітектурі RISC покращує ефективність роботи процесора?
8. Як процесори Intel Core i9 реалізують архітектуру x86-64 і в чому їх особливості?
9. Які принципи лежать в основі архітектури процесорів AMD Ryzen 9, і чим вона відрізняється від Intel Core i9?
10. Як архітектура ARM Cortex і процесори Apple M4 використовують принципи RISC для досягнення енергоефективності та високої продуктивності?

ТЕМА 10. КОНВЕЄРНЕ ВИКОНАННЯ КОМАНД

Етапи конвеєризації. Багатоступеневі конвеєри для обробки потоків інструкцій.

Етапи конвеєризації

Конвеєр команд – це компонент центрального процесора, призначений для прискорення виконання машинних команд шляхом суміщення певних стадій їх виконання в часі.

В основі концепції конвеєризації обчислень є твердження про те, що процес обробки машинної команди можна розбити на декілька практично незалежних етапів, які потім можна суміщати в часі для декількох команд в відповідній апаратурі (конвеєрі команд).

Загальноприйнятим в теорії конвеєрних структур є така послідовність етапів: вибірка, декодування, інтерпретація та відправлення команди до відповідного операційного пристрою в залежності від різновиду операції, виконання, звертання до пам'яті та збереження результату обчислень в регістрі.

Слід зазначити, що така структура є типовою для RISC-архітектур, семантика та кількість етапів в яких зазвичай не дуже відрізняється від наведених, тоді як в CISC-архітектурах конвеєр може бути набагато складнішим.

В процесі виконання команд в конвеєрі, відбувається суміщення вказаних етапів виконання для декількох команд, тобто, в конвеєрі одночасно знаходяться декілька команд на різних стадіях. Кожний етап конвеєрного обчислення виконується за один машинний цикл. Зрозуміло, що виконання однієї окремої команди на послідовному процесорі, де команда виконується зразу за один машинний цикл, і може бути швидшим, аніж в конвеєрній організації. За рахунок суміщення виконання різних команд, загальна швидкодія істотно збільшується.

Спочатку визначимо декілька понять для розуміння типів конвеєрів і принципів їх організації.

Пропускна здатність – це максимальна кількість команд, які виконуються за один такт машинного часу (Instructions Per Cycle, IPC).

Тривалість етапу – це кількість машинних циклів для виконання одного етапу конвеєрного обчислення і вона може бути різною для різних етапів;

Необхідний (максимальний) ступінь паралелізму – це кількість незалежних команд, які потрібно завантажити в конвеєр для повного використання його потенціалу.

Багатоступеневі конвеєри для обробки потоків інструкцій
Найпростіший класичний конвеєр має структуру, як на рисунку 10.1.

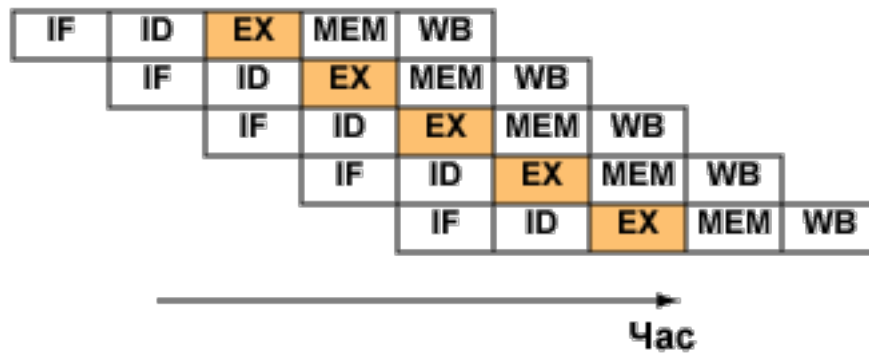


Рисунок 10.1 – Схема класичного конвеєра [15]

Характеристики пристрою:

- пропускна здатність – 1 IPC;
- тривалість стадії – 1 цикл;
- необхідний максимальний ступінь паралелізму – 1 команда.

Максимальний ступінь паралелізму у такій моделі обмежується одночасною обробкою лише однієї інструкції на кожній стадії, тобто в різних стадіях одночасно може перебувати кілька інструкцій, але кожна окремо виконується в рамках своєї стадії.

Щоб підтримувати таку ідеальну роботу конвеєра, необхідно мінімізувати структурні, інформаційні та керуючі залежності між командами. Будь-які збої, такі як залежності між даними або непередбачувані переходи, можуть призводити до зупинок конвеєра (столів) і зниження його ефективності. Тому класичний конвеєр добре працює в умовах, коли потік інструкцій є передбачуваним і добре впорядкованим.

Якщо класичний конвеєр обробляє одну інструкцію на стадію в кожен цикл, то **суперскалярний** підхід (рис. 10.2) дозволяє завантажувати та виконувати дві, три або більше інструкцій паралельно. Це досягається завдяки наявності кількох виконавчих пристроїв і складної логіки розподілу інструкцій між ними

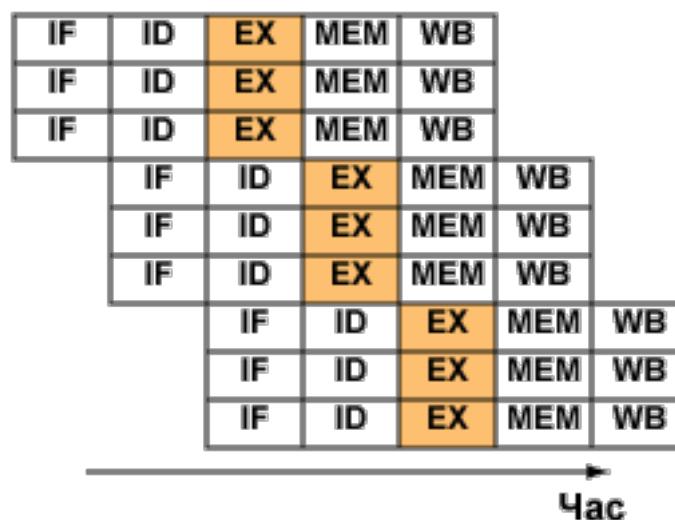


Рисунок 10.2 – Схема суперскалярного конвеєра [15]

Суть суперскалярного конвеєра полягає у динамічному аналізі інструкцій на наявність залежностей та їх одночасному відправленні на виконання, якщо це дозволяє логіка програми.

Використання суперскалярного конвеєра дозволяє значно збільшити пропускну здатність процесора, наблизивши її до кількох IPC (наприклад, 2–4 інструкції за такт і більше). Це особливо важливо для сучасних високопродуктивних процесорів, які мають обробляти великі обсяги даних і виконувати численні паралельні завдання у реальному часі.

Характеристики пристрою:

- пропускну здатність – n IPC (3 IPC на рисунку);
- тривалість стадії – 1 цикл;
- необхідний максимальний ступінь паралелізму – n команд (на рисунку їх 3).

Як бачимо, в тому випадку, коли тривалість всіх стадій є однаковою, пропускну здатність суперскалярного конвеєра в n разів вища за звичайний. Для повного завантаження та максимальної ефективності необхідно, щоби на кожному циклі роботи він був завантажений n командами, які можуть виконуватись незалежно одна від одної. Така вимога істотно обмежує ефективність суперскалярного конвеєра, адже ступінь схованого паралелізму звичайних послідовних програм, зазвичай, є не дуже високою і в середньому обмежена числами близько 2.5-3 IPC, хоча на окремих класах задач може бути істотно вищою.

Для задач, де критично важливо досягти максимальної тактової частоти процесора і збільшити продуктивність обробки за рахунок більш дрібної розбивки виконання інструкцій уперконвеєр використовується **суперконвеєр** (рис. 10.3).

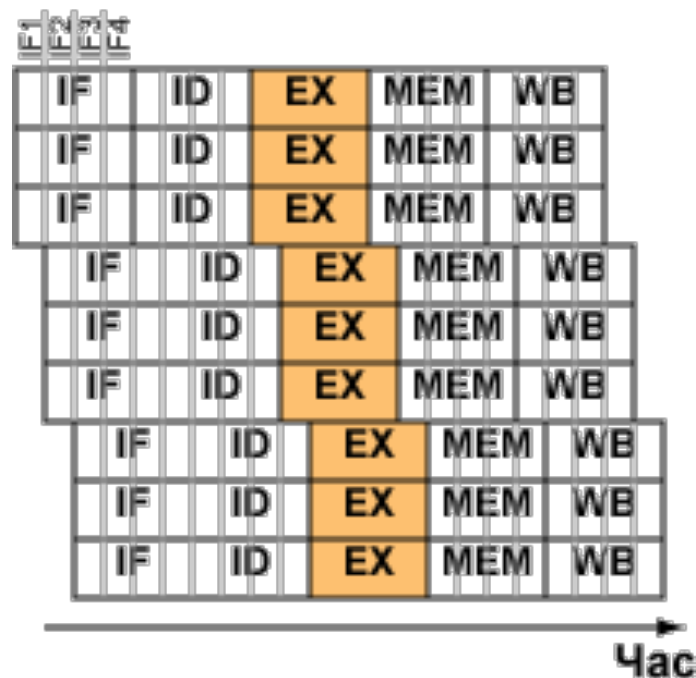


Рисунок 10.3 – Схема суперконвеєра [15]

Головна перевага суперконвеєра полягає в збільшенні частоти процесора без радикальної зміни самої логіки обробки інструкцій.

Припустимо, що деякі з вказаних нами етапів типового 5-стадійного конвеєра в свою чергу також допускають розбиття на елементарніші «суперстадії». Тоді стає можливим зменшення тривалості машинного циклу за рахунок того, що кожна з таких «елементарних» стадій може виконуватись швидше аніж «звичайна», а в результаті проміжок часу, вимірюваний одним машинним циклом, між виходом оброблених команд з конвеєра зменшується. В теоретичній моделі з малюнку, де кожна зі «звичайних» стадій розбита на 4 «суперстадії», отримаємо 4-кратний приріст швидкодії за умови, що кількість часу на виконання «звичайної» стадії залишається незмінною.

Таким чином, в загальному випадку для не суперскалярного суперконвеєра, в якому кожній «звичайній» стадії відповідає m «суперстадій», або елементарних стадій:

- пропускна здатність – 1 IPC, але тривалість циклу $1/m$ в порівнянні зі звичайним конвеєром;
- тривалість стадії – 1 цикл для «суперстадії», або m циклів для кожної «звичайної стадії»;
- необхідний максимальний ступінь паралелізму – m команд.

Однак нескінченно дрібнити стадії конвеєра та підвищувати таким чином тактову частоту неможливо. Довгі конвеєри стають вкрай неефективними при виконанні команд, які змінюють хід виконання програми (тобто команд умовних та безумовних переходів). Ці команди за певних умов призводять до необхідності повного перезавантаження конвеєра, а чим більше стадій, тим більше часу для цього необхідно [15].

Контрольні питання:

1. Які основні етапи конвеєризації обробки інструкцій у процесорі?
2. У чому полягає принцип роботи багатоступеневого конвеєра?
3. Як поділ інструкцій на стадії впливає на ефективність конвеєризації?
4. Які типові стадії містить класичний п'ятиступеневий конвеєр?
5. Що таке структурні, інформаційні та керуючі конфлікти в багатоступеневих конвеєрах?
6. Як багатоступеневий конвеєр допомагає збільшити пропускну здатність процесора?
7. Які виклики виникають при збільшенні кількості стадій у конвеєрі і як їх долають сучасні процесори?

ТЕМА 11. СПОСОБИ АДРЕСАЦІЇ ОПЕРАНДІВ

Режими адресації та їх вплив на продуктивність процесора. Вибір способів адресації операндів.

Режими адресації та їх вплив на продуктивність процесора

Формат команди з полями для розпізнавання типу адресації будується так, щоб процесор міг чітко визначити, яку саме операцію потрібно виконати і як обробити операнди (рис. 11.1).

КОП	ТА1	A1	ТА2	A2
-----	-----	----	-----	----

Рисунок 11.1 – Формат команди з полями адресацій [11]

Зазвичай, команда складається з кількох обов'язкових полів: поля коду операції, яке вказує тип виконуваної операції; полів для зазначення операндів або їхніх адрес; і окремого поля або біту, що визначає тип адресації, тобто спосіб інтерпретації операндів.

За способом інтерпретації розрізняють такі типи адресації операндів: пряма адресація, непряма адресація, регістрова адресація.

Поле коду операції керує вибором необхідної арифметичної, логічної або іншої операції, тоді як поля операндів можуть містити безпосередні значення (літерали), номери регістрів або вказівки на пам'яттєві адреси. Тип адресації зазвичай визначається через спеціальні біти або комбінації в складі формату команди, які повідомляють процесору, чи потрібно, наприклад, звертатися безпосередньо до пам'яті, обробляти адресу через регістр чи застосовувати певне зміщення.

Такий розподіл полів у команді дозволяє забезпечити гнучкість і універсальність обробки інструкцій, адаптуючись під різні типи задач. У складних процесорах, особливо CISC, формат команд може бути дуже різноманітним і мати змінну довжину, тоді як у процесорах типу RISC команди зазвичай фіксованої довжини та спрощеного формату для швидшого декодування і ефективного розпаралелювання виконання.

Безпосередня адресація – це спосіб, при якому операнд, необхідний для виконання команди, міститься прямо в самій команді. Інакше кажучи, замість того, щоб вказувати адресу комірки пам'яті або регістра, процесор отримує конкретне значення безпосередньо з поля операнда команди. Це дозволяє значно прискорити виконання інструкції, оскільки немає потреби здійснювати додаткові звернення до пам'яті для отримання даних. Безпосередня адресація особливо ефективна в тих випадках, коли потрібно працювати з константами або фіксованими даними, наприклад, додати число 5 до вмісту регістра або встановити певне значення в пам'ять. При цьому команда стає самодостатньою для виконання операції, що спрощує логіку обробки і знижує загальне навантаження на систему пам'яті (рис. 11.2).



Рисунок 11.2 – Безпосередня адресація [11]

Головним обмеженням безпосередньої адресації є те, що розмір безпосереднього операнда обмежений довжиною інструкції, оскільки частина бітів зарезервована для коду операції та інших полів. Тому безпосередня адресація зазвичай використовується для невеликих за розміром констант або при спрощеній обробці даних.

На відміну від безпосередньої, **пряма адресація** пам'яті (рис. 11.3) є зручною для роботи з великими обсягами даних, які не поміщаються у регістровому файлі, тоді як пряма адресація регістрів забезпечує максимальну швидкість виконання інструкцій за рахунок роботи з невеликим, але дуже швидким обсягом даних усередині процесора.

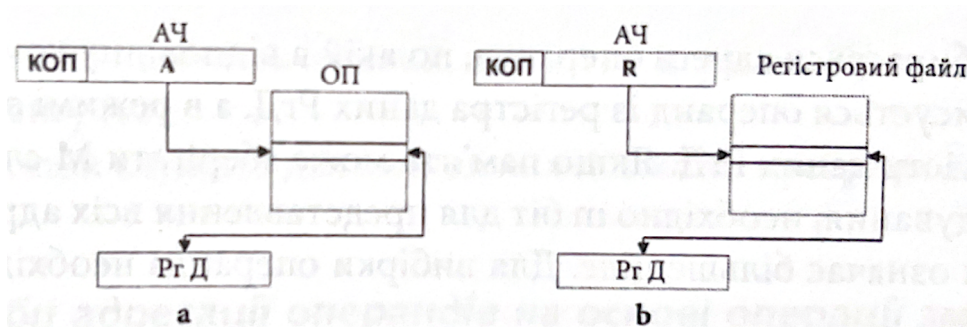


Рисунок 11.3 – Пряма адресація:

а) основної пам'яті; б) регістрового файлу процесора [11]

Пряма адресація основної пам'яті (а) означає, що команда містить у своєму полі адресу тієї комірки оперативної пам'яті, де безпосередньо розташовані дані. Під час виконання інструкції процесор зчитує цю адресу з команди, звертається до відповідної комірки пам'яті і використовує отримане значення як операнд. Це простий і зрозумілий спосіб організації звернень до даних, але він має недолік у вигляді затримок, оскільки доступ до оперативної пам'яті повільніший порівняно із доступом до регістрів.

Пряма адресація регістрового файлу процесора (б) працює за аналогічним принципом, тільки замість звернення до пам'яті вказується номер регістра всередині процесора, де зберігається потрібний операнд. Завдяки малій затримці доступу до регістрів цей тип адресації є набагато швидшим і більш ефективним. Пряма регістрова адресація широко використовується в сучасних архітектурах, особливо в процесорах з архітектурою RISC, де основний акцент робиться на опрацюванні даних всередині регістрів без частих звернень до оперативної пам'яті.

Непряма адресація основної пам'яті, яка вимагає здійснення двох звернень (рис. 11.4), використовується в тих випадках, коли в полі команди вказується не сама адреса потрібних даних, а адреса комірки пам'яті, де

зберігається справжня адреса операнда. При виконанні такої інструкції процесор спочатку звертається до пам'яті за вказаною у команді адресою і отримує відтуда нову адресу, після чого здійснює друге звернення за отриманою адресою, щоб отримати або змінити сам операнд.

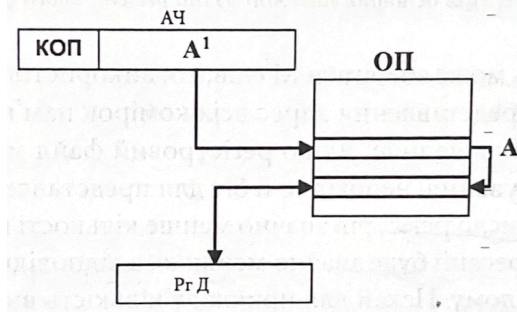


Рисунок 11.4 – Непряма адресація основної пам'яті, яка вимагає здійснення двох звернень [11]

Такий механізм дозволяє підвищити гнучкість програм, оскільки місце розташування фактичних даних можна змінювати динамічно без потреби змінювати саму інструкцію. Однак вартість цього підходу полягає в додаткових затримках через подвійне звернення до пам'яті, що може помітно впливати на загальну продуктивність, особливо в системах з повільною оперативною пам'яттю.

Непряма адресація основної пам'яті з використанням регістрового файлу процесора (рис. 11.5) полягає у тому, що команда не містить безпосередньо ані даних, ані їхньої адреси, а містить посилання на регістр процесора, у якому зберігається адреса необхідного операнда в пам'яті.

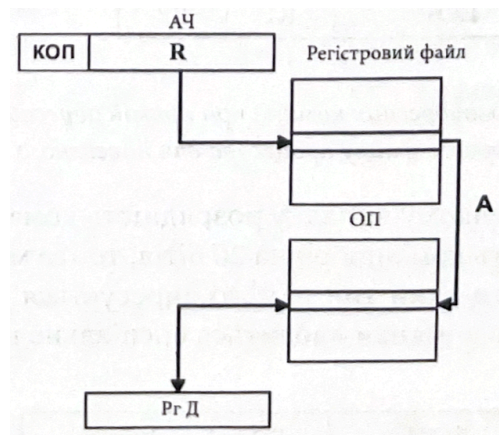


Рисунок 11.5 – Непряма адресація основної пам'яті з використанням регістрового файлу процесора [11]

При виконанні такої інструкції процесор спочатку зчитує значення з зазначеного регістра, розглядає це значення як адресу і потім звертається до відповідної комірки пам'яті для отримання або запису даних.

Такий спосіб адресації суттєво прискорює доступ у порівнянні з непрямою пам'яттєвою адресацією через пам'ять, оскільки перше звернення здійснюється до регістрового файлу, доступ до якого набагато швидший за доступ до оперативної пам'яті. Водночас непряма регістр-пам'ятна адресація

дозволяє створювати гнучкіші програми, оскільки зміна вмісту реєстра автоматично змінює, куди саме буде здійснюватися звернення в пам'яті.

Цей метод особливо корисний для реалізації вказівників, обробки масивів, стеків або організації складних динамічних структур даних у програмах. Крім того, він широко використовується в сучасних процесорах для оптимізації роботи циклів та процедур із часто змінними адресами даних.

Вибір способів адресації операндів

Вибір способів адресації операндів у процесорі визначається характером виконуваної задачі, вимогами до продуктивності, гнучкості програми та ефективності використання ресурсів пам'яті (таблиця 11.1).

Таблиця 11.1 – Основні аргументи вибору способів адресації операндів

Спосіб адресації	Місце розміщення операнда
Безпосередня	Значення операнда знаходиться в полі адресної частини команди
Пряма	Адреса операнда знаходиться в полі адресної частини команди
Непряма	Адресна частина команди вказує адресу розміщення операнда
Базова	Виконавча адреса формується шляхом додавання до вмісту реєстра бази зміщення із поля адресної частини команди
Відносна	Виконавча адреса формується шляхом додавання вмісту адресного поля команди до вмісту програмного лічильника
Індексна	Виконавча адреса формується шляхом додавання вмісту адресного поля команди до вмісту спеціального реєстра
Неявна	Адреса операнда в явному вигляді в команді відсутня

Отже, якщо необхідно швидко обробляти невеликі константи або прості значення, найбільш ефективною буде безпосередня адресація, оскільки вона не потребує додаткових звернень до пам'яті. Для роботи з даними, які часто змінюються або зберігаються в реєстрах, обирається пряма регістрова адресація – вона забезпечує мінімальну затримку доступу до операндів.

Частота використання різних способів адресації в комп'ютерах залежить від багатьох факторів: типу архітектури, вимог до параметрів комп'ютера, характеру вирішуваних задач. За час існування обчислювальної техніки була створена велика кількість комп'ютерів, у яких застосовано різноманітні способи адресації та їх поєднання. Тому дати однозначну відповідь стосовно вибору найкращого набору способів адресації практично неможливо.

Контрольні питання:

1. Що таке спосіб адресації операндів і яка його роль у виконанні команд процесора?

2. У чому полягає суть безпосередньої адресації і для яких задач вона найефективніша?
3. Як працює пряма адресація основної пам'яті і які її переваги та недоліки?
4. Які особливості прямої адресації через регістровий файл процесора?
5. Що таке непряма адресація і чому вона вимагає двох звернень у випадку з основною пам'яттю?
6. Як реалізується непряма адресація з використанням регістрів і в яких випадках вона переважно застосовується?
7. Які чинники впливають на вибір способу адресації операндів?

ТЕМА 12. ОРГАНІЗАЦІЯ ВВЕДЕННЯ-ВИВЕДЕННЯ

Керування перериваннями: пріоритет переривань, конвеєризація переривань. Оптимізація доступу до даних: синхронізація кешу, особливості використання ресурсів пам'яті. Розширення інтерфейсів.

Керування перериваннями: пріоритет переривань, конвеєризація переривань

Переривання (interrupt) – це сигнал, що повідомляє процесор про настання якої-небудь події, яка потребує невідкладної уваги. При цьому виконання поточної послідовності команд призупиняється і керування передається обробнику переривання, який реагує на подію та обслуговує її, після чого повертає управління в перерваний код.

У класичних способах програмно-керованого введення-виведення передбачалась передача одного байта за один раз. Кероване перериваннями введення-виведення могло маніпулювати лише даними по одному байту за один раз або малими блоками, залежно від виду пристрою, що бере участь у введенні-виведенні. Така схема керування застосовувалась у ранніх архітектурах процесорів, як то Intel P5, AMD K7 тощо.

У сучасних процесорах механізми введення-виведення стали набагато складнішими і ефективнішими, порівняно з початковими моделями передачі даних по одному байту. Хоча програмно-кероване введення-виведення та кероване перериваннями введення-виведення залишаються концептуально актуальними, в сучасних системах акцент робиться на мінімізацію участі процесора в обробці потоків даних. Пристрої введення-виведення часто працюють із великими об'ємами даних завдяки використанню буферизації і підтримці передач блоками, що дозволяє передавати одразу сотні або навіть тисячі байтів без активної участі центрального процесора.

В комп'ютерах використовують слідує чотири методи керування введенням-виведенням:

1. Програмно кероване введення-виведення;
2. Кероване перериваннями введення-виведення ;
3. Прямий доступ до пам'яті;
4. Введення-виведення під керуванням периферійних процесорів.

Оскільки в комп'ютері може бути багато пристроїв введення-виведення, при зверненні до них їх потрібно розпізнати. Спосіб розпізнавання залежить від способу їх підключення до процесора. Так як в комп'ютері може бути багато пристроїв введення-виведення, то при зверненні до них їх потрібно розпізнати. Методи розпізнавання залежить від способів їх підключення до процесора. Якщо пристрої введення-виведення підключені до процесора за допомогою окремої шини введення-виведення (рис. 21.1), то в цьому випадку процесор звертається до пристрою введення-виведення за його номером.

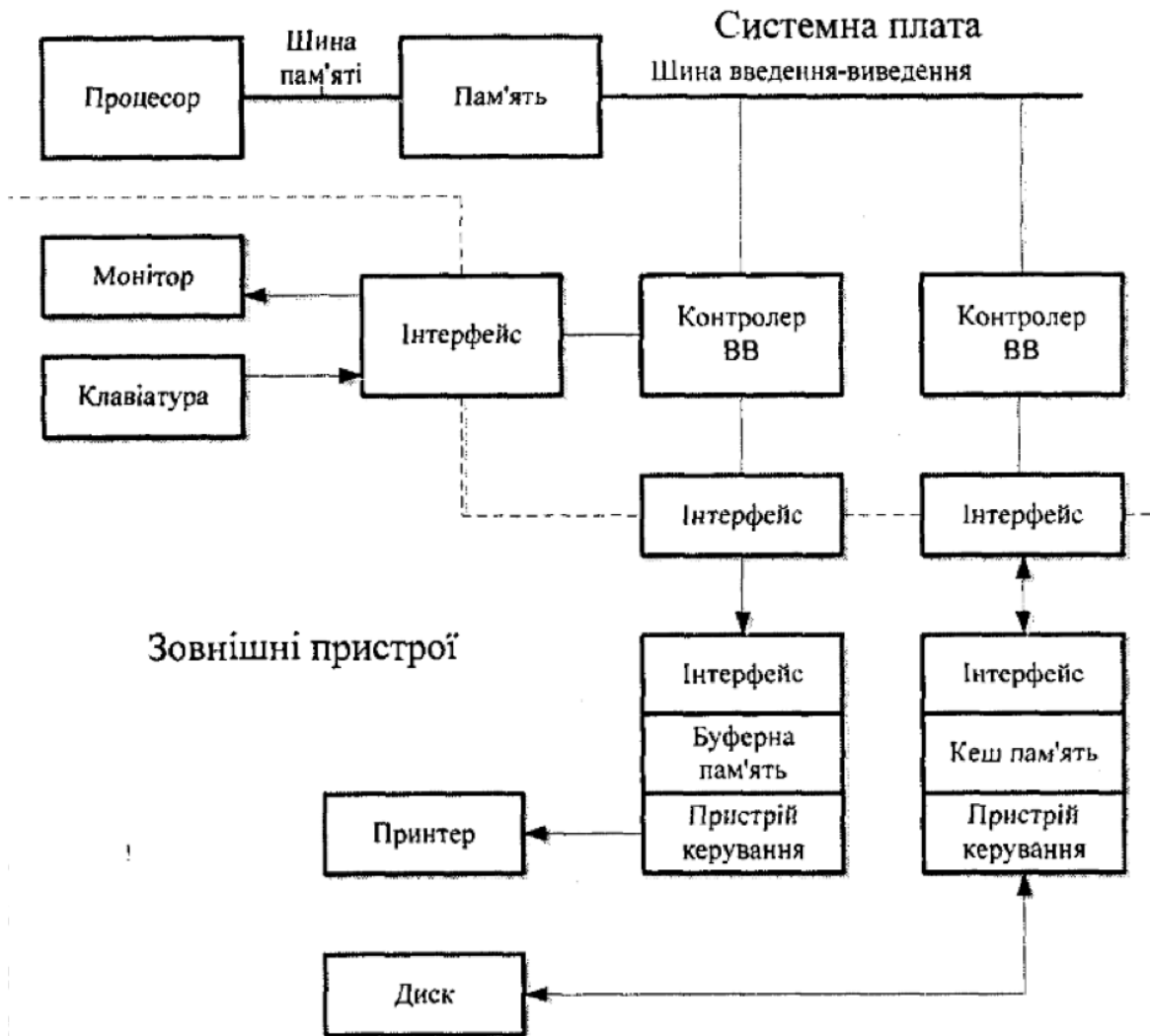


Рисунок 12.1 – Система введення-виведення із шинним інтерфейсом [16]

Процесор у таких випадках ініціює операцію, передає команду відповідному контролеру введення-виведення, а потім або переходить до виконання інших задач, очікуючи сигнал переривання про завершення операції, або використовує спеціальні механізми, наприклад, прямий доступ до пам'яті (DMA). Прямий доступ до пам'яті дозволяє пристроям введення-виведення самостійно здійснювати обмін даними із пам'яттю без навантаження процесора, що істотно підвищує загальну пропускну здатність системи.

Контролер переривань обслуговує запити переривань від зовнішніх пристроїв за допомогою процедур переривань. Контролер приймає запит, визначає пріоритет цього запиту та визначає права конкретного зовнішнього пристрою на його обслуговування. Він посилає у мікропроцесор сигнал переривання та повідомляє його про номер або адресу обслуговуваного пристрою або про адресу першої команди підпрограми оброблення цього запиту. Мікропроцесор призупиняє виконання поточної програми та виконує підпрограму обслуговування переривання. Після завершення підпрограми мікропроцесор повертається до виконання перерваної програми.

Оптимізація доступу до даних: синхронізація кешу, особливості використання ресурсів пам'яті

У сучасних процесорах вибір ємності кеш-пам'яті – це завжди баланс між швидкістю, вартістю та ефективністю. Кеш повинен бути досить невеликим, щоб не надто підвищувати загальну вартість системи, і водночас достатньо великим, щоб більшість звернень до пам'яті обслуговувалися швидше саме через кеш. Із збільшенням об'єму кешу зростає складність адресації та затримки доступу, навіть за однакових технологій виробництва.

Реальна ефективність кешу залежить від характеру завдань: при зростанні об'єму кількість промахів у кеші помітно зменшується, але після певної межі цей ефект стає майже непомітним. Для більшості прикладних задач оптимальним вважається кеш у межах від 1 до 512 Кбайт, хоча сучасні процесори мають набагато більші багаторівневі кеші. Важливим чинником також є розмір рядка кешу: спочатку його збільшення знижує кількість промахів завдяки принципу локальності, але надто великі рядки навпаки погіршують ефективність через зменшення кількості доступних рядків і більшу ймовірність непотрібного завантаження даних.

Тому в проектуванні кеш-пам'яті завжди враховується компроміс між обсягом, часом доступу, вартістю та статистикою використання даних, що дозволяє адаптувати процесор під різні типи навантажень і досягати максимальної продуктивності в реальних умовах.

Оптимізація доступу до даних у сучасних процесорах тісно пов'язана із синхронізацією кешу, яка забезпечує узгодженість між даними в кеш-пам'яті та основній пам'яті. Оскільки кеш працює як проміжний буфер з набагато швидшим доступом, ніж оперативна пам'ять, він активно використовується для зберігання часто потрібних даних. Проблема виникає тоді, коли той самий блок даних може змінюватися в різних кешах кількох ядер або у зовнішній пам'яті, тому потрібно постійно слідкувати за їх актуальністю та узгодженістю.

Синхронізація кешу відбувається через спеціальні протоколи когерентності, наприклад, MESI (Modified, Exclusive, Shared, Invalid), які визначають стан кожного кешованого блока даних. Якщо одне ядро змінює дані, інші ядра повинні або отримати оновлену копію, або позначити свої копії як застарілі. Цей процес відбувається за допомогою обміну сигналами між кешами через спеціальні шини або мережеві структури в багатоядерних процесорах, що дозволяє підтримувати єдину послідовну картину пам'яті для всіх ядер.

Крім когерентності, оптимізація також включає техніки попереднього вибіркового завантаження, організацію багаторівневих кешів (L1, L2, L3) і адаптивне управління кеш-лініями для мінімізації затримок. Завдяки цьому сучасні процесори можуть дуже ефективно працювати навіть із великими обсягами даних, зберігаючи баланс між швидкістю доступу і забезпеченням коректності обробки інформації.

У таблиці 12.1 наведено типові ємності кеш-пам'яті в архітектурах комп'ютерів Intel, AMD та Apple 2025 року.

Таблиця 12.1 – Ємності кеш-пам'яті мікропроцесорних десктопних архітектур

Процесор (архітектура)	Кеш L1 (на ядро)	Кеш L2 (на ядро)	Кеш L3 (загальний)
Intel Core i9 (Meteor Lake)	80–128 Кбайт	2 Мбайт	24–36 Мбайт
AMD Ryzen 9 (Zen 4)	64 Кбайт	1–2 Мбайти	64 Мбайти
Apple M4 (ARM)	128 Кбайт	2–4 Мбайти	32–64 Мбайти

Усі наведені процесори використовують адаптивні механізми для оптимізації роботи кешу. Це включає в себе автоматичне очищення кешу (при заповненні) або механізми "недавнього використання", коли менш часто використовувані дані витісняються з кешу для звільнення місця для більш актуальних. Крім того, вони активно використовують принципи локальності даних, зокрема, локальність за часом (дані, які часто використовуються, ймовірно, будуть використовуватися знову) і локальність за місцем (дані, що використовуються разом, ймовірно, будуть використовуватися поруч). Це дозволяє зберігати найбільш "гарячі" дані в кеші для швидкого доступу. Кеш третього рівня (L3) часто є спільним для всіх ядер і має більшу ємність, ніж L1 і L2, що дозволяє знижувати кількість промахів. У деяких архітектурах (наприклад, у серверних процесорах) є також кеш L4, що служить ще більшим буфером для даних.

Розширення інтерфейсів.

Розширення інтерфейсів у сучасних комп'ютерних архітектурах стосується процесу вдосконалення способів обміну даними між процесором, пам'яттю та зовнішніми пристроями для підвищення загальної продуктивності системи. Зі зростанням обчислювальних потужностей і обсягів даних класичні інтерфейси стали обмеженням, тому виникла потреба у створенні нових або вдосконаленні існуючих стандартів. Розширення відбувається за рахунок збільшення пропускної здатності (наприклад, переходу з PCIe 3.0 на PCIe 5.0 і 6.0), зменшення затримок, підвищення енергоефективності та інтеграції більшої кількості каналів обміну даними одночасно.

Сучасні процесори і чипи активно використовують високошвидкісні шини і контролери пам'яті, що підтримують нові стандарти типу DDR5, LPDDR5X або HBM3, які суттєво збільшують швидкість доступу до оперативної пам'яті. Крім того, в архітектуру часто вбудовуються власні високошвидкісні інтерфейси для взаємодії між компонентами, такі як AMD Infinity Fabric або Apple UltraFusion, що забезпечують тісну інтеграцію багатьох обчислювальних блоків із мінімальними втратами швидкодії.

Розвиток інтерфейсів також включає підтримку нових технологій введення-виведення, таких як Thunderbolt 4, USB4, NVMe для швидких SSD-

накопичувачів та спеціалізовані протоколи обробки великих обсягів даних для штучного інтелекту і графічних процесорів. Усе це дозволяє комп'ютерним системам працювати зі складними завданнями в реальному часі, забезпечуючи високу масштабованість і стабільність у багатозадачних середовищах.

Контрольні питання:

1. Що таке переривання в архітектурі процесора і для чого воно використовується?
2. Як визначається пріоритет переривань у багатозадачних системах?
3. Що таке маскування і немаскування переривань та в чому їх відмінність?
4. Яким чином організовується конвеєризація обробки переривань у сучасних процесорах?
5. Які проблеми можуть виникати при конвеєризації переривань і як їх вирішують?
6. Що таке синхронізація кешу і чому вона критична в багатоядерних процесорах?
7. Як протоколи когерентності кешу типу MESI забезпечують правильність обробки даних?
8. Які особливості ефективного використання ресурсів пам'яті в сучасних комп'ютерах?
9. Чому важливо враховувати принцип локальності при оптимізації доступу до даних?
10. Що таке розширення інтерфейсів у процесорних системах і які проблеми воно вирішує?
11. Які особливості мають сучасні інтерфейси передачі даних, наприклад PCIe 5.0 чи Thunderbolt 4?
12. Як впливає інтеграція власних високошвидкісних інтерфейсів типу Infinity Fabric чи UltraFusion на продуктивність системи?

ТЕМА 13. ПАРАЛЕЛІЗМ ТА БАГАТОПОТОЧНІСТЬ

Паралелізм на рівні даних і команд, таксономія Фліна. Технології багатопоточності в сучасних процесорах. Технології віртуалізації ядер.

Паралелізм на рівні даних і команд, таксономія Фліна

Сьогодні використання принципів паралельної обробки інформації в архітектурі комп'ютерів є звичним. Практично всі сучасні комп'ютери використовують той або інший вид паралельної обробки інформації. На презентаціях нових мікросхем і в інформаційних матеріалах виробників це подається як останнє слово техніки і передовий край науки, і це дійсно так, якщо розглядати реалізацію цих принципів у мініатюрних рамках одного кристалу.

Водночас, ідеї паралельної обробки інформації з'явилися досить давно. Вони впроваджувалися в найбільш передових, дорогих, а тому одиничних, комп'ютерах свого часу. Пізніше, після належного відпрацювання технології і здешевлення виробництва, вони використовувалися в комп'ютерах середнього класу, і нарешті сьогодні вони в повному обсязі втілюються у робочих станціях і персональних комп'ютерах.

Потрібно відзначити, що збільшення кількості процесорів у багатопроцесорній системі не завжди пропорційно зменшує час вирішення задачі. Це явище пояснюється законом Амдала, який визначає теоретичне обмеження прискорення обчислень при паралельній обробці. Згідно з цим законом, максимальне прискорення обмежується частиною задачі, яка не піддається паралелізації, тобто виконується послідовно. Навіть при нескінченному збільшенні кількості процесорів, час виконання послідовної частини залишається незмінним, що створює межу для загального прискорення.

Таксономія Флінна (англ. Flynn's taxonomy) – це загальна класифікація архітектур за ознаками наявності паралелізму в потоках команд і даних. Була запропонована 1966 року М. Флінном. Суть полягає у тому, що уся розмаїтість архітектур в цій таксономії зводиться до чотирьох класів, тобто, Флінн виділяє чотири класи архітектур: SISD, SIMD, MISD, MIMD.

SISD (single instruction – single data) – одиночний потік команд і даних. Паралелізм відсутній. До цієї категорії відносяться послідовні архітектури, у тому числі й фон-нойманівського типу.

SIMD (single instruction – multiple data) – одиночний потік команд і декілька потоків даних. Паралелізм в таких архітектурах полягає в можливості одночасного виконання однієї й тої ж операції над декількома елементами даних. Досягається це централізованою видачею команд декільком обчислювальним пристроям спільним для них пристроєм управління. Найпоширенішими представниками архітектур типу SIMD є так звані векторні

комп'ютери, оптимізовані для паралельного виконання однотипних операцій над елементами векторів і матриць. Спеціалізовані векторні процесори іноді вбудовуються в комп'ютери загального призначення. Зокрема, в багатьох сучасних мікропроцесорах вбудовані обмежені можливості векторних обчислень для обробки мультимедіа.

MISD (multiple instruction – single data) – декілька потоків команд, одиночний потік даних. Фахівці дотепер не прийшли до єдиної думки щодо того, які архітектури відносити до даного класу. Деякі включають у нього конвеєрні обчислювачі, в яких цілісна операція розбивається на послідовність простіших етапів з суміщенням різних етапів для різних порцій даних у часі. Таким чином, в певному кожна порція даних від моменту завантаження з пам'яті до запису результату обробки проходить через декілька етапів обробки (стадій конвеєра). Подібні принципи покладені в основу систолічних архітектур, які також належать до цього класу.

MIMD (multiple instructions – multiple data) – декілька потоків команд і даних. Це найпоширеніший на сьогодні клас паралельних архітектур, у яких кожний процесор здатний працювати незалежно від інших над своїм завданням.

Архітектурні особливості комп'ютерних систем описано з погляду потоку команд (інструкцій) та потоку даних. Такий підхід дає змогу відносити архітектури комп'ютерів до одного з чотирьох класів (табл. 13.1). Класифікація здійснена з погляду не структури, а того, як у комп'ютері його машинні команди взаємодіють з даними.

Таблиця 13.1 – Класифікація архітектур комп'ютерних систем (за Фліном)

Потік команд	Одиничний потік даних (ОД)	Множинний потік даних (МД)
Одиничний (ОК)	ОКОД (SISD) (однопроцесорні комп'ютери)	ОКМД (SIMD) (комп'ютери з паралельними або асоціативними процесорами)
Множинний (МК)	МКОД (MISD) (конвеєрні магістральні комп'ютери)	МКМД (MIMD) (багатопроцесорні або багатомашинні комплекси)

У схемах архітектур зазвичай присутні ПУ (пристрій управління), АЛП і шини для передачі інструкцій та даних. Ось опис чотирьох основних класів з поясненням схематичної будови. Кожен тип у класифікації Фліна має своє місце в сучасному обчислювальному середовищі: від серійних однокористувацьких задач до масивно паралельних обчислень у кластерних системах, високопродуктивних обчисленнях (HPC) та штучному інтелекті. Розуміння цієї класифікації важливе для вибору оптимальної архітектури для конкретної задачі чи системи (рис. 13.1).

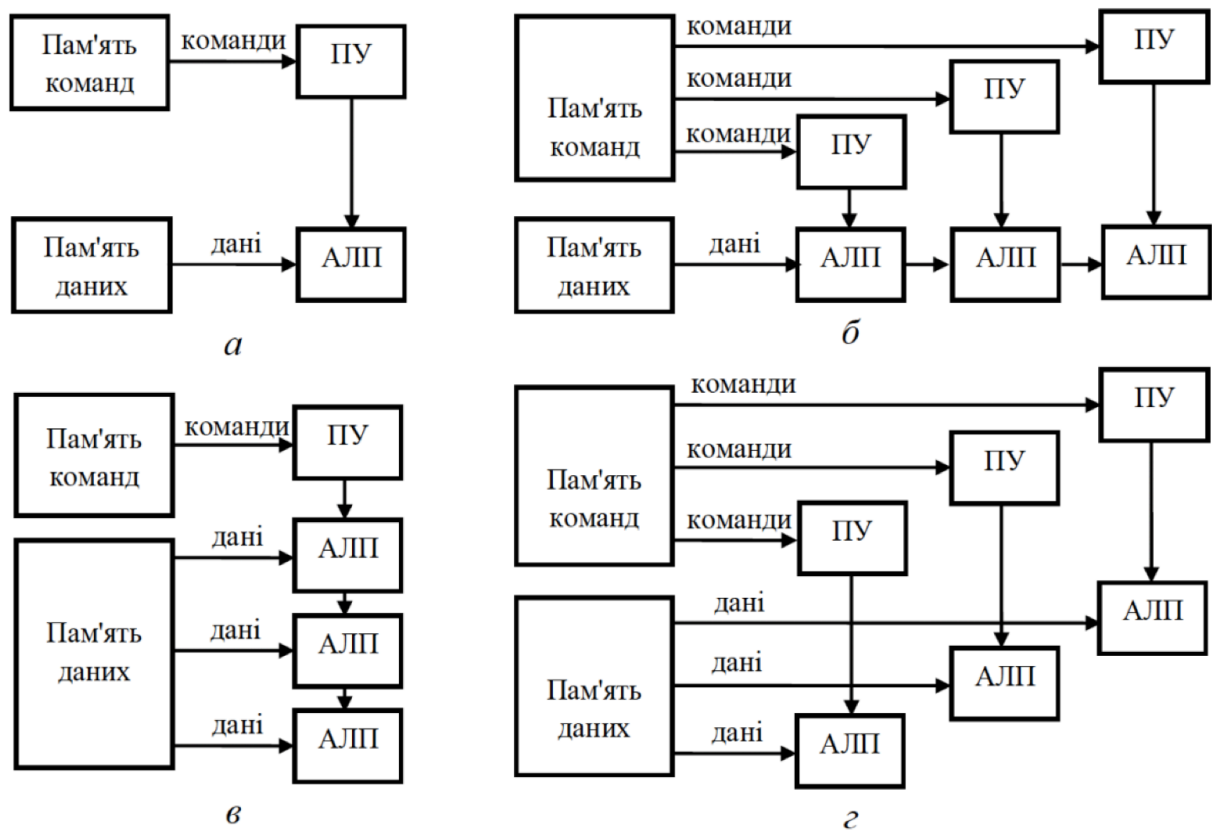


Рисунок 13.1 – Архітектура комп'ютерних систем за Флінном: а – SISD; б – MISD; в – SIMD; г – MIMD

Класифікація архітектур за Фліном залишається фундаментальною основою для розуміння типів обчислювальних систем і їхньої організації. Вона допомагає розмежувати, як саме інструкції та дані обробляються всередині системи, що особливо важливо при проектуванні високопродуктивних і паралельних обчислювальних рішень.

Технології багатопоточності в сучасних процесорах

Багатопоточність – це підхід у програмуванні, який дозволяє виконувати кілька потоків виконання в одному процесі. Потік – найменша одиниця обробки в операційній системі, і багатопоточність дозволяє використовувати паралельні обчислення підвищення продуктивності комп'ютера.

Багатопотокові програми виконують кілька завдань одночасно, що призводить до більш ефективного використання ресурсів. Це особливо корисно на багатоядерних процесорах, де кожне ядро виконує свій потік. Програми можуть відповідати на введення користувача або виконувати фонові завдання, не блокуючи основний потік виконання. Це робить програми більш чуйними і покращує користувацький досвід. Розбиття великих завдань на дрібніші та їх виконання паралельно дозволяє прискорити загальний час. Це корисно в обчислювально-інтенсивних задачах, таких як обробка даних або графіка.

Багатопоточні програми більш ефективно використовують ресурси комп'ютера (процесорний час і пам'ять), зменшуючи простой в очікуванні завершення операцій. Асинхронне програмування зменшує час очікування

введення/виводу та підвищує загальну продуктивність. Однак, слід зазначити, що багатопоточність також може внести складнощі у розробку програм та управління ресурсами. Програмістам необхідно акуратно обробляти синхронізацію та доступ до загальних ресурсів, щоб уникнути гонок даних та інших проблем, пов'язаних із паралельним виконанням.

У сучасних процесорах Intel, AMD та Apple технології багатопоточності стали фундаментальними для підвищення продуктивності та енергоефективності, особливо в умовах зростання вимог до обробки великих обсягів даних і виконання складних багатозадачних сценаріїв.

Розглянемо найбільш популярні технології багатопоточності у виробників сучасних процесорів.

Intel застосовує технологію **Hyper-Threading** (HT), яка дозволяє кожному фізичному ядру одночасно обробляти два потоки інструкцій. Завдяки цьому підвищується ефективність використання виконавчих блоків ядра: коли один потік зупиняється через очікування даних з пам'яті, інший потік може продовжувати використовувати процесорний ресурс. Hyper-Threading істотно покращує продуктивність у багатопотокових додатках, таких як віртуалізація, моделювання та обробка відео.

AMD реалізує технологію **Simultaneous Multithreading** (SMT), яка аналогічно дозволяє кожному фізичному ядру обслуговувати два потоки. У сучасних архітектурах Zen 3 і Zen 4 SMT є невід'ємною частиною стратегії максимально ефективного використання всіх обчислювальних ресурсів ядра. Завдяки вдалій архітектурній оптимізації SMT у AMD забезпечує особливо високу ефективність у робочих навантаженнях, що добре масштабуються, таких як серверні обчислення, рендеринг і наукові розрахунки.

Apple, у своїх процесорах серії M (наприклад, M2, M3, M4), використовує власну підхід до багатопоточності. Архітектура ARM-базованих процесорів Apple поділяє ядра на високопродуктивні performance cores і енергоефективні efficiency cores. Кожне ядро виконує по одному потоку, проте загальна система оптимізована так, що високопродуктивні ядра обробляють важкі завдання, а енергоефективні – фонові і менш критичні процеси. Це дозволяє ефективно балансувати навантаження, зменшуючи енергоспоживання без втрати швидкодії. Фактично Apple досягає схожого ефекту багатопоточності через комбінацію багатоядерності і розумного розподілення потоків між різними типами ядер.

Технології віртуалізації ядер

Технології віртуалізації ядер – це методи та технології, які дозволяють ефективно використовувати ресурси багатоядерних процесорів шляхом створення віртуальних або логічних ядер, що можуть виконувати завдання незалежно одне від одного. Це дозволяє збільшити продуктивність системи, покращити ефективність обчислень і забезпечити паралельне виконання кількох потоків або операцій.

У багатоядерних процесорах фізичні ядра можуть бути поділені на віртуальні або логічні ядра, кожне з яких здатне виконувати окремі інструкції

або потоки. Це дозволяє досягти більшої продуктивності при обробці паралельних завдань, особливо в багатозадачних або обчислювальних навантаженнях.

При апаратній підтримці віртуалізації апаратне забезпечення надає архітектурну підтримку, що полегшує побудову гіпервізорів і дозволяє гостьовим ОС бути запущеними в ізоляції. Апаратна підтримка віртуалізації була вперше представлена на IBM System/370 в 1972 році, для використання разом з операційною системою VM/370.

Компанії Intel і AMD представили технології апаратної віртуалізації архітектури x86. Sun Microsystems (нині Корпорація Oracle) додали схожі особливості у своїх процесорах UltraSPARC T-серії. Прикладами платформ віртуалізації адаптованих до такого апаратного забезпечення відносяться KVM, VMware Workstation, VMware Fusion, Hyper-V, Windows Virtual PC, Xen Desktop для Parallels Mac, Oracle VM сервер для SPARC, VirtualBox і Parallels Workstation.

Контрольні питання:

1. Що таке паралелізм на рівні даних і як він реалізується в сучасних процесорах?
2. Які основні типи паралелізму на рівні команд і як вони впливають на продуктивність процесорів?
3. Як класифікується паралелізм за таксономією Фліна, і що включають в себе кожен з її типів?
4. Як таксономія Фліна допомагає зрозуміти різницю між архітектурами з одночасним виконанням інструкцій та з паралельною обробкою даних?
5. Що таке багатопоточність і чим вона відрізняється від традиційного одноядерного виконання?
6. Як технологія Intel Hyper-Threading впливає на ефективність багатозадачності в сучасних процесорах Intel?
7. Які переваги і недоліки має технологія багатопоточності в процесорах AMD?
8. Як технологія віртуалізації ядер дозволяє покращити ефективність використання ресурсів у багатоядерних процесорах?
9. Як працює технологія віртуалізації ядер на прикладі процесорів Intel та AMD?
10. У чому різниця між віртуалізацією ядер і використанням логічних ядер у багатоядерних процесорах?
11. Яким чином багатопоточність і віртуалізація ядер допомагають в зниженні енергоспоживання при високих навантаженнях?
12. Які сучасні технології віртуалізації процесорів використовуються для покращення паралельного виконання в обчислювальних системах?

ТЕМА 14. АРХІТЕКТУРИ КЛАСТЕРІВ ТА СУПЕРКОМП'ЮТЕРІВ

Високопродуктивні обчислення. Керування енергоспоживанням у сучасних процесорах. Роль архітектури в штучному інтелекті та машинному навчанні. Розгляд апаратних засобів захисту в сучасних CPU (Intel, AMD, Apple) – захист конфіденційних обчислень і шифрування пам'яті для віртуалізації.

Високопродуктивні обчислення

Архітектура кластерів та суперкомп'ютерів є основою для розв'язання надскладних обчислювальних задач, що виникають у наукових, інженерних та промислових галузях. Завдяки розвитку технологій паралельної обробки даних та високопродуктивних обчислень, ці системи стали незамінними для таких задач, як моделювання кліматичних змін, геномні дослідження, розрахунки в астрономії, фізиці та біології, а також обробка великих даних у фінансових і медичних сферах. Кластери та суперкомп'ютери дозволяють досягати надзвичайно високої продуктивності, шляхом використання великої кількості процесорів, що працюють паралельно, і спеціалізованих технологій для ефективного обміну даними між компонентами системи.

Завдяки своїй архітектурі, яка включає численні обчислювальні вузли, спільні та розподілені пам'яті, а також високошвидкісні мережі для комунікацій, ці системи дозволяють вирішувати надскладні задачі, що потребують величезної обчислювальної потужності та великих обсягів пам'яті. Сучасні суперкомп'ютери є прикладом такого інженерного досягнення, яке дозволяє отримувати точні результати в найкоротші терміни. Важливим аспектом таких систем є їх гнучкість і можливість для масштабування, що дозволяє пристосовувати системи до зростаючих вимог і потреб у різних сферах наукової та технічної діяльності.

Кластер – це група взаємопов'язаних комп'ютерів (вузлів), які працюють разом як єдине ціле, щоб обробляти великі обсяги даних або виконувати складні обчислення. Кластери можуть бути різного типу залежно від завдань, які вони виконують.

Попередні обчислювальні функції підходили для вирішення простих проблем, але вони не здатні синхронізувати сучасні технології та досягнення. Сценарії ціноутворення, низька продуктивність, надійність і деякі інші причини не могли отримати доступ до керування величезними даними, а попередні обчислювальні системи не можуть вирішити сучасні ІТ-проблеми. У рамках цього ІТ-відділи запровадили інноваційний підхід у вирішенні обмежень однієї системи, а саме «кластерні обчислення». Переваги цієї технології дозволили багатьом організаціям впровадити цю технологію у свою діяльність. І сьогодні популярність цього змусила нас обговорити ці поняття. Давайте розберемося з цією технологією. Що таке кластерні обчислення? Кластерні обчислення означає, що багато комп'ютерів, підключених до

мережі, працюють як єдине ціле. Кожен комп'ютер, підключений до мережі, називається вузлом.

Кластерні обчислення пропонують рішення для вирішення складних проблем, забезпечуючи високу швидкість обчислення та покращену цілісність даних. Підключені комп'ютери виконують операції разом, створюючи враження єдиної системи – віртуальної машини. Цей процес називається прозорістю системи [16].

Базуючись на принципі розподілених систем, ця мережева технологія виконує свої операції. Тут LAN є блоком підключення і цей процес визначається, як прозорість системи. Кластерні обчислення мають такі особливості: усі підключені комп'ютери є машинами одного типу; вони тісно з'єднані через виділені мережеві з'єднання; усі комп'ютери мають спільний домашній каталог. Апаратна конфігурація кластерів відрізняється залежно від вибраних мережевих технологій (рис. 14.1).

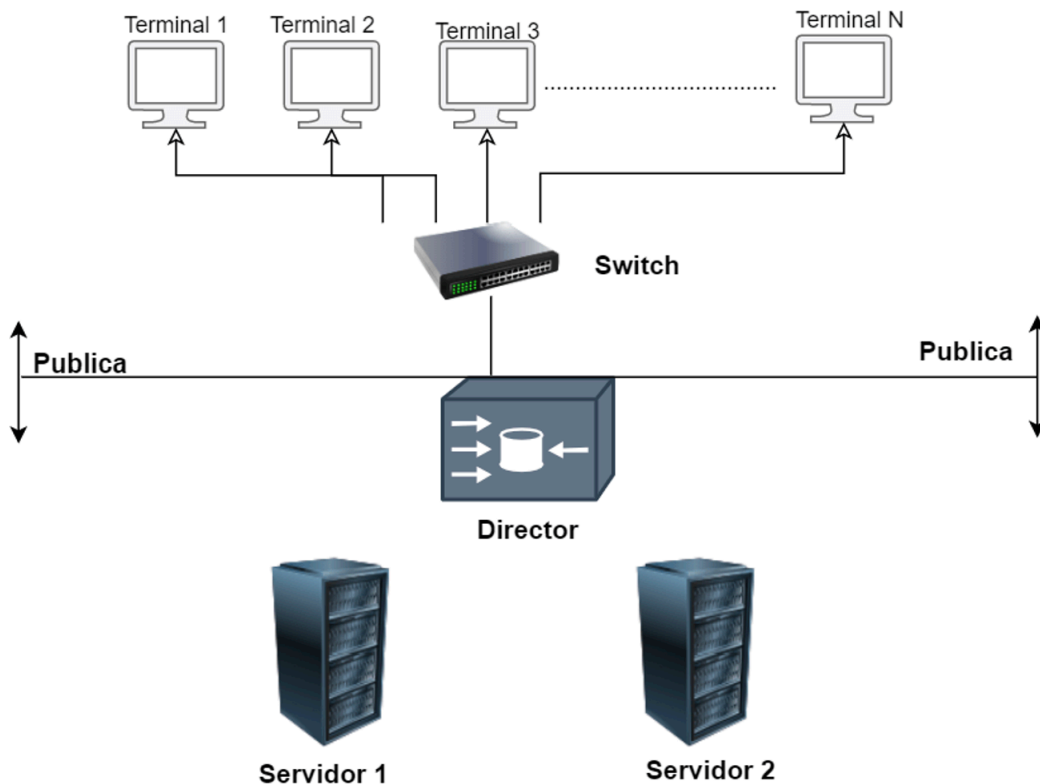


Рисунок 14.1 – Архітектура кластера [16]

Система поєднує як відкриті, так і закриті кластери, де відкриті кластери потребують IP-адреси для всіх вузлів, доступ до яких здійснюється лише через Інтернет або Інтернет. Цей тип кластеризації викликає підвищені проблеми безпеки. У закритій кластеризації вузли приховані за вузлом шлюзу та забезпечують підвищений захист.

Суперкомп'ютери з їхніми швидкісними центральними процесорами призначені для обчислень і вирішення багатьох інших обчислювальних завдань значно швидше, ніж стандартні потужні ПК. Вони використовують високорівневий паралелізм для збільшення швидкості виконання операцій.

Раніше суперкомп'ютери були призначені для дослідження та розробки різних видів зброї з метою забезпечення державної безпеки, а також у різних теоретичних проектах. Останні досягнення у потужності, характеристиках та продуктивності показують, наскільки ефективно вони були обладнані новою технікою та здатні справлятися зі складними завданнями масового виробництва та відповідати вимогам споживачів.

Суперкомп'ютери позиціонуються як найшвидші комп'ютери із усіх існуючих. Останнє покоління суперкомп'ютерів здатне виконувати мільйон мільярдів обчислень за секунду або 1 петафлоп (Petaflop). Подібні силові та технічні можливості спонукають наукові, урядові та дослідницькі організації створювати та розвивати надзвичайно складні наукові, інженерні та комерційні проекти, проводити математичні обчислення, що потребують переробки великої кількості даних, проводити експерименти, що відтворюють умови фізичної реальності, і вирішувати складні технічні завдання.

Дивлячись на стан останніх років, існують суперкомп'ютери, які можуть виконувати до ста квадриліонів флопів (flops – це одиниця виміру продуктивності комп'ютерів). Поширена сфера застосування суперкомп'ютерів. В основному суперкомп'ютери використовувалися військовими та розвідувальними службами, університетами та незалежними дослідницькими центрами, а також елітними інженерними корпораціями. Тепер нове покоління дешевших суперкомп'ютерів знаходить свою нішу серед приватного сектору та сфер ділової активності. Ці високопродуктивні суперкомп'ютери пропонують гарантовану потужність та витончені технічні можливості без особливих витрат та складної системи, як у попередніх версіях. Вони використовуються в таких сферах, як обчислювальна гідродинаміка, електронний дизайн, нафтогазопошукові роботи, довгострокові прогнози погоди, дослідження ядерної енергії, метеорологія та структурний аналіз.

Сучасні суперкомп'ютери – це дуже потужні обчислювальні системи. Швидкість їхньої роботи вже досягає десяти і більше петафлопс. Хоча буквально 3-5 років тому йшлося про терафлопс. Наразі вже йдеться про створення першого екзафлопного (10¹⁸ операцій на секунду) комп'ютера. Потужність обчислювальної техніки у наші дні збільшується з разючою швидкістю. Щоправда, фізики також не відстають і ставлять завдання відповідної складності. Останнє десятиліття дало дивовижний стрибок у вирішенні найскладніших завдань.

Керування енергоспоживанням у сучасних процесорах

Керування енергоспоживанням у сучасних процесорах є критично важливою складовою їхньої архітектури, оскільки дозволяє збалансувати продуктивність і теплові характеристики при збереженні енергоефективності. З огляду на те, що сучасні процесори мають мільярди транзисторів і працюють на високих тактових частотах, їхня здатність адаптувати споживання енергії до поточного навантаження є вирішальною для мобільних пристроїв, ноутбуків і навіть серверів.

Сучасні CPU від Intel, AMD і Apple активно використовують такі технології, як динамічне масштабування частоти Dynamic Frequency Scaling, коли ядра процесора працюють на нижчій частоті під час простою або легких завдань, та підвищують її при зростанні навантаження (Intel Turbo Boost, AMD Precision Boost, Apple Performance Cores). Крім того, застосовується динамічне керування напругою (Dynamic Voltage Scaling), яке ще більше знижує споживання, особливо при роботі з невеликими навантаженнями.

Також впроваджуються глибокі стани сну C-States, коли ядра або окремі блоки процесора повністю відключаються від живлення в режимах очікування. У багатоядерних процесорах можлива деактивація окремих ядер Core Parking, що дозволяє суттєво знизити загальне споживання. Виробники інтегрують у процесори спеціальні контролери енергоспоживання, які працюють незалежно від основних ядер, аналізують поточні сценарії використання і оптимізують живлення в реальному часі. Усе це формує сучасний підхід до енергоефективних обчислень, де енергоспоживання стало не менш важливим показником, ніж обчислювальна потужність.

Наприклад, у сучасних архітектурах Intel, AMD та Apple вбудовані цілісні системи керування енергоспоживанням, які оптимізовані під специфіку їхніх процесорів.

Архітектура Intel Alder Lake та Intel Meteor Lake. На цій архітектурі процесори підтримують технологію Intel Dynamic Tuning, яка в реальному часі адаптує частоту та напругу залежно від температури, навантаження та типу задачі. Вони мають розподіл на енергоефективні ядра E-Cores та високопродуктивні P-Cores, що дозволяє, наприклад, запускати фонові завдання на E-Cores, зменшуючи енергоспоживання без втрати продуктивності. Також використовуються технології Speed Shift, Turbo Boost Max 3.0 і підтримка C-States до рівнів C10.

Архітектура AMD Zen 4. На цій архітектурі процесори використовують Precision Boost 2, яка дозволяє динамічно піднімати частоту окремих ядер у межах TDP, і Curve Optimizer, що забезпечує тонке налаштування напруги на рівні окремих ядер. Функція CPPC2 Collaborative Power Performance Control у тандемі з ОС дозволяє процесору самому вирішувати, коли і які ядра активувати, зменшуючи втрати енергії. У Zen 4 додатково реалізовані стани глибокого сну і зниження напруги між тактами Power Gating.

Архітектура Apple M-серії, зокрема M2, M3. Чипи Apple Silicon мають чітке розділення на efficiency cores та performance cores, які використовуються залежно від навантаження. Контролер живлення повністю інтегрований в SoC і приймає рішення автономно від ОС. Вони також реалізують fine-grained clock gating та Dynamic Voltage and Frequency Scaling на рівні кожного обчислювального блока. Завдяки цій оптимізації MacBook-и на Apple M-серії можуть працювати без активного охолодження при низьких навантаженнях, зберігаючи потужність на вимогу.

Ці приклади демонструють різні підходи до енергоефективності, що відображає філософію кожного виробника: Intel орієнтується на гнучкість, AMD – на масштабовану динаміку, Apple – на інтеграцію і баланс.

Роль архітектури в штучному інтелекті та машинному навчанні

Архітектура комп'ютерів відіграє ключову роль у розвитку штучного інтелекту та машинного навчання, оскільки саме вона визначає, наскільки ефективно можна обробляти великі обсяги даних і виконувати мільйони паралельних обчислень. ШІ-системи, особливо нейронні мережі, потребують високої пропускної здатності пам'яті, широких векторних обчислень і спеціалізованих обчислювальних блоків. Класична архітектура CPU не справляється з такими навантаженнями настільки ефективно, як графічні процесори GPU, тензорні процесори TPU чи інші прискорювачі.

У сучасних ШІ-застосуваннях важливу роль відіграє **паралелізм**, що реалізується на апаратному рівні за допомогою архітектур SIMD, раніше згаданих у темі 13 цього конспекту лекцій у архітектурі Фліна, а також завдяки обчислювальним кластерам з тисячами вузлів. Спеціалізовані архітектури, наприклад NVIDIA Hopper або Google TPU, мають тисячі ядер, оптимізованих під матричні обчислення, які є основою для навчання глибоких нейронних мереж. Це дає змогу в десятки разів пришвидшити як навчання, так і інференс моделей.

Крім того, сучасні процесори підтримують апаратне прискорення для ML навіть на рівні інструкцій, таких як AVX-512 у Intel чи SIMD-розширення у Apple Silicon). Розвиток архітектури комп'ютерів у напрямі енергоефективності, масштабованості та гетерогенності дозволяє створювати системи, здатні обробляти ШІ-навантаження від невеликих вбудованих пристроїв до великих суперкомп'ютерів у хмарі.

Розгляд апаратних засобів захисту в сучасних CPU (Intel, AMD, Apple) – захист конфіденційних обчислень і шифрування пам'яті для віртуалізації

У сучасних процесорах провідних виробників – Intel, AMD та Apple – активно впроваджуються апаратні засоби захисту, що дозволяють забезпечити конфіденційність даних навіть під час виконання коду в середовищі з обмеженою довірою, як-от віртуальні машини чи хмарна інфраструктура. Це критично важливо в умовах сучасної кібербезпеки, особливо для ізольованого виконання, шифрування пам'яті та захисту від атак на рівні гіпервізора або ОС.

Intel впровадила технологію Intel SGX (Software Guard Extensions), яка дозволяє створювати ізольовані області пам'яті – так звані *енклави*, де можна виконувати чутливі обчислення, недоступні навіть для операційної системи чи гіпервізора. Крім того, Intel TME (Total Memory Encryption) забезпечує апаратне шифрування всієї оперативної пам'яті, що ускладнює атаки через доступ до фізичних модулів пам'яті.

AMD розробила більш широку архітектуру для захисту пам'яті під назвою **SEV (Secure Encrypted Virtualization)**. Вона дозволяє кожній віртуальній машині мати окремий ключ для шифрування пам'яті, завдяки чому навіть гіпервізор не може читати її вміст. Удосконалення SEV-ES і SEV-SNP додатково захищають реєстри процесора та забезпечують перевірку цілісності гостьових середовищ.

Apple, використовуючи архітектуру ARM, застосовує апаратні механізми на рівні SoC, такі як **Secure Enclave** – окремий безпечний співпроцесор для обробки конфіденційної інформації, зокрема біометричних даних та криптографічних ключів. Також у Apple Silicon реалізоване шифрування пам'яті в реальному часі, що не потребує програмної підтримки й майже не впливає на продуктивність.

Ці апаратні засоби захисту стали фундаментальними елементами довірених обчислень, дозволяючи створювати безпечні середовища для хмарного ШІ, цифрового підпису, фінансових операцій та зберігання персональних даних.

Контрольні питання:

1. Які основні принципи побудови кластерних систем і як вони забезпечують масштабованість обчислень?
2. Чим відрізняється архітектура суперкомп'ютера від звичайного багатоядерного комп'ютера?
3. Які типи зв'язків між вузлами кластерів використовуються у високопродуктивних обчислювальних системах?
4. Як система керування задачами працює в архітектурі кластерів?
5. У чому полягає мета енергозберігаючих технологій у сучасних процесорах?
6. Які апаратні засоби використовуються для зниження енергоспоживання у процесорах Intel?
7. Як технології AMD забезпечують баланс між продуктивністю та енергоефективністю?
8. Як архітектурні особливості процесорів впливають на ефективність навчання нейронних мереж?
9. Які архітектурні рішення використовуються в GPU і TPU для прискорення ML-обчислень?
10. Яку роль відіграє кешування та пропускна здатність пам'яті в обчисленнях ШІ?
11. У чому полягає принцип роботи технології Intel SGX для захисту даних?
12. Як реалізується ізоляція і шифрування пам'яті в AMD SEV?
13. Яке призначення Secure Enclave у процесорах Apple і як він забезпечує конфіденційність?
14. Як апаратні засоби захисту впливають на безпеку віртуалізованих середовищ?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Поповський С. Історія розвитку комп'ютерної техніки. *Друкарня*. URL: <http://surl.li/xsrijc> (дата звернення: 01.06.2024).
2. Ковальчук О. Як IBM створила програмістів, 2024. *Cargofy Inc*. URL: <https://www.youtube.com/watch?v=wArY0FuDXvQ> (дата звернення: 01.06.2024).
3. Демиденко М., Руденко О. Навчальний посібник з дисципліни «Комп'ютерна схемотехніка та архітектура комп'ютерів» для студентів спеціальності 122 «Комп'ютерні науки». Полтава : НУПП, 2023. 203 с. URL: <http://surl.li/qtkzeb> (дата звернення: 03.06.2024).
4. Архітектура комп'ютера. *Вікіпедія*. URL: <http://surl.li/rtaiqi> (дата звернення: 01.06.2024).
5. Цирульник С. Проектування мікропроцесорних систем: ел. навч. посібник. <https://ekt.elit.sumdu.edu.ua>. URL: <http://surl.li/wdvtlz> (дата звернення: 02.06.2024).
6. Вовк П. Архітектура комп'ютерів : Електрон. навч.-метод. посіб. Луцьк, 2022. URL: <http://surl.li/hyrnad> (дата звернення: 04.06.2024).
7. Особливості платформи Intel LGA1851: пам'ять DDR5, підтримка PCIe 5.0 і Thunderbolt 4. *Overclockers.ua*. URL: <https://bit.ly/4cO8R1E> (дата звернення: 17.03.2025).
8. Матвієнко С. Протоколи та стандарти. *Інформаційний ресурс про електроніку та програмування*. URL: <http://surl.li/dlazwt> (дата звернення: 01.06.2024).
9. Diuzhakova D. Central Processing Unit: Hardware Component Processing Data. *Intechhouse*. URL: <http://surl.li/iwrgsn> (date of access: 01.06.2024).
10. Тиртишніков О. Архітектура комп'ютера. Класифікація та покоління обчислювальних систем. Основні показники та характеристики комп'ютерів. *ppt-online.org*. URL: <https://ppt-online.org/751973> (дата звернення: 15.03.2025).
11. Черняшук Н.Л., Христинець Н.А. Архітектура комп'ютерів. Конспект лекцій для здобувачів першого (бакалаврського) рівня вищої освіти освітньо-професійної програми «Комп'ютерна інженерія» галузі знань 12 «Інформаційні технології» спеціальності 123 Комп'ютерна інженерія денної та заочної форм навчання. Луцьк : Луцький НТУ, 2020. 108 с.
12. Difference Between RISC and CISC - Tpoint Tech. *www.tpointtech.com*. URL: <https://www.tpointtech.com/risc-vs-cisc> (date of access: 28.04.2025).
13. He Y., Chen X. Survey and Comparison of Pipeline of Some RISC and CISC System Architectures. *2023 8th International Conference on Computer and Communication Systems (ICCCS)*, Guangzhou, China, 21–23 April 2023. 2023. URL: <https://bit.ly/431cZYZ> date of access: 28.04.2025).
14. Спасюк Д. Огляд процесора Intel Core i9-14900K – новий король продуктивності. *IT-community*. URL: <https://bit.ly/4lQysuZ> (дата звернення: 19.03.2025).

15. Учасники проєктів Вікімедіа. Конвеєр команд – Вікіпедія. *Вікіпедія*. 2020. URL: https://uk.wikipedia.org/wiki/Конвеєр_команд (дата звернення: 21.02.2025).
16. Ходукін М. Архітектура комп'ютера: процес введення-виведення. URL: <https://www.duet.edu.ua/uploads/DocS/10st2.pdf> (дата звернення: 15.01.2025).
17. Mitton L. Amdahl's Law: Understanding the Basics | Splunk. *Splunk*. URL: https://www.splunk.com/en_us/blog/learn/amdahls-law.html(date of access: 28.04.2025).

Архітектура комп'ютерів: конспект лекцій для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерна інженерія» галузі знань 12 «Інформаційні технології» спеціальності 123 Комп'ютерна інженерія денної та заочної форм навчання / уклад. Н.А. Христинець. Луцьк: ЛНТУ, 2025. 112 с.

Комп'ютерний набір:
Редактор

Н.А.Христинець
Н.А.Христинець

Підп. до друку «__»_____ 2025р.
Формат 60x84/16. Папір офс. Гарнітура Таймс.
Ум. друк. арк. _____. Тираж ____ прим. Зам. _____

Відділ іміджу та промоції
Луцького національного технічного університету
43018, м. Луцьк, вул. Львівська, 75