

**ФАХОВИЙ КОЛЕДЖ РАКЕТНО-КОСМІЧНОГО МАШИНОБУДУВАННЯ
ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ
імені Олеся Гончара**

**Конспект лекції
з дисципліни
«Архітектура комп'ютерів»**

**Дніпро
2023**

Зміст

Тема 1.1 Поняття архітектури і структури комп'ютера. Архітектура фон Неймана. Ієрархічний принцип побудови апаратних та програмних засобів комп'ютера.	3
Тема 1.2 Типи сучасних комп'ютерів. Основні характеристики комп'ютерів.	10
Тема 1.3 Форм-фактори материнської плати. Компоненти системної плати.	20
Тема 2.1 Функції і склад мікропроцесорів. Архітектури мікропроцесорів.	30
Тема 2.2 Організація і функціонування системи пам'яті.	36
Тема 2.3 Накопичувачі на жорсткому магнітному диску. Зовнішні запам'ятовуючі пристрої.	40
Тема 2.4 Відеоадаптери і монітори. Аудіосистема комп'ютера.	47
Тема 3.1 Подання чисел у комп'ютерах. Обробка чисел із плаваючою комою.	56
Тема 3.2 Сегментована пам'ять і регістрова структура мікропроцесора i8086.	61
Тема 3.3 Система команд процесорів.	68
Тема 3.4 Організація взаємодії процесора з основною пам'яттю через кеш пам'ять. Віртуальна пам'ять.	72
Тема 4.1 Рівні паралелізму. Класифікація архітектури комп'ютерних систем.	83
Тема 4.2 Обчислювальні системи класу SIMD та MIMD. Багатомашинні та багатопроцесорні комп'ютерні системи.	89
Тема 4.3 Ефективність обчислювальних систем.	97

Тема 1.1 Поняття архітектури і структури комп'ютера. Архітектура фон Неймана. Ієрархічний принцип побудови апаратних та програмних засобів комп'ютера.

Лекція 1 (2 год.) Історичні аспекти розвитку комп'ютерів. Функції та основні функціональні вузли комп'ютера.

План лекції:

1. Основні етапи розвитку комп'ютерної техніки.
2. Функції комп'ютера.
3. Основні функціональні вузли комп'ютера.

В 1946 році *Джон фон Нейман* (John von Neumann) описав архітектуру комп'ютера, яка є найпоширенішою і в даний час. В своїй науковій роботі він назвав новий пристрій обчислювальним інструментом (computing instrument), звідки логічно появилася сучасна назва комп'ютера. Ця архітектура дістала назву *архітектури Джона фон Неймана*, хоча в зарубіжній літературі частіше використовується термін "Instruction Set Architecture" ("архітектура системи команд"), що вказує як на рівень опису архітектури комп'ютера, так і на програмний принцип його роботи. До перших розробників та впроваджувачів цієї архітектури належать *Чарльз Бебідж* (Charles Babbage), конструктор двох механічних комп'ютерів – різницевої та аналітичної машини (1822-1833 pp.), *Джон Атанасов* (John Atanasoff) – автор першого спеціалізованого комп'ютера ABC (1939 рік), *Конрад Цузе* (Konrad Zuse) (1936-1944 pp.) та *Ховард Айкен* (Howard Aiken) (1941-1946 pp.) конструктори перших електромеханічних комп'ютерів відповідно Z1-Z4 та Mark1-Mark2; *Алан Тюрінг* (Allan Turing) – один з розробників першого електронного комп'ютера Colossus (1943 рік), *Преспер Екерт* (Presper Eckert) та *Джон Моучлі* (John Mauchly) – розробники першого універсального комп'ютера ENIAC (1946 рік), який був описаний в науковій роботі Джона фон Неймана, *Мауріс Уїлкс* (M. Wilkes) – розробник проекту комп'ютера EDSAC (1946 рік).

До перших електронних комп'ютерів з програмним керуванням належить і мала електронно-обчислювальна машина (МЕОМ), створена в 1951 році в Інституті електротехніки Академії Наук України, м. Київ під керівництвом академіка *С. А. Лебедева*.

Сучасний стан технології проектування комп'ютерів базується на теоретичних роботах ряду видатних дослідників, в першу чергу *С. Крея*, засновника фірми Cray Research, яка випустила кілька поколінь найпотужніших в світі комп'ютерів; *М. Фліна*, який розробив найуживанішу класифікацію комп'ютерів; *Г. Амдаля*, який сформував ряд новітніх положень розвитку комп'ютерів; *Р. Томасуло*, який запропонував підхід до вирішення питання ефективного завантаження комп'ютера; *Д. Коука*, який запропонував архітектуру "Америка", на основі якої збудовано комп'ютер з спрощеною системою команд RS/6000 фірми IBM (усі вони є вихідцями з цієї фірми);

Д. Хеннессі (Стенфордський університет), автора архітектури комп'ютера з спрощеною системою команд MIPS, засновника фірми MIPS (зараз це підрозділ фірми Silicon Graphics, однієї з провідних фірм по виготовленню суперкомп'ютерів); Д. Паттерсона (Каліфорнійський університет), автора архітектури комп'ютерів з спрощеною системою команд RISC I та RISC II, технології RAID збереження даних тощо; їх ідеї були апробовані шляхом створення та практичного використання комп'ютерів фірм Cray Research, IBM, Silicon Graphics, Intel, MIPS, SUN Microsystems, Digital Equipment Corporation, Transmeta та багатьох інших.

Комп'ютер представляє собою електронний пристрій, який містить *апаратні засоби і програмне забезпечення* та автоматично, відповідно до програми, виконує алгоритм вирішення заданої задачі.

Під *алгоритмом* розуміють точний припис, що задає обчислювальний процес вирішення задачі, а під *задачею* – сформульоване намагання отримати з множини вхідних даних і початкових умов та з множини можливих вихідних даних підмножину вихідних даних, що повністю задовольняють початкові умови і вхідні дані. Алгоритм характеризується множиною параметрів вхідних, проміжних та вихідних даних, правилом вводу даних, правилом початку, правилом обробки даних, правилом закінчення, правилом виводу даних. Для виконання алгоритму комп'ютер приймає вхідну інформацію в цифровій формі, обробляє її відповідно до вказівок команд програми виконання алгоритму, та видає результати обчислень.

До **основних функцій, які виконує комп'ютер**, належать наступні:

- сприйняття вхідної інформації – вхідних даних, які підлягають обробці, та програм вирішення задач (програм обробки вхідних даних);
- зберігання інформації, тобто вхідних і проміжних даних та результатів обчислень, програм вирішення задач, довідникової інформації, програм операційної системи комп'ютера і т. д.;
- виконання арифметичних, логічних та інших операцій;
- автоматичне керування роботою складових частин комп'ютера, їх взаємодією між собою та з зовнішніми пристроями відповідно до програми;
- виведення результатів обчислень.

Для забезпечення виконання цих функцій до складу комп'ютера повинні входити такі основні функціональні вузли: пристрої введення інформації, пристрої виведення інформації, пам'ять та процесор.

Коротко зупинимось на функціях та складі кожного вузла комп'ютера.

Пристрої введення-виведення виконують введення та виведення інформації. До числа пристроїв введення-виведення належать:

- пристрої введення – клавіатура, миша, сканер, відеокамера і т. д.;
- пристрої виведення – монітори (з електронно-променевою трубкою та рідкокристалічними), принтер, графопобудовувач і т. д.

Пам'ять призначена для зберігання інформації. Типи пам'яті:

– кеш пам'ять (КП) – високошвидкісна пам'ять невеликої ємності, використання якої дозволяє прискорити обмін інформацією між основною пам'яттю і процесором.

– основна або оперативна пам'ять (ОП) – пам'ять великої ємності, яка зберігає інформацію, що підлягає обробці в процесорі. До інформації, яка зберігається в ОП, належать вхідні дані, які підлягають обробці відповідно до виконуваного алгоритму, результати проміжних обчислень, вихідні дані, команди програми виконання алгоритму, активна частина операційної системи комп'ютера. Інформація в ОП постійно змінюється, тобто це пам'ять для короткотермінового зберігання інформації;

– зовнішня пам'ять (ЗП) – пам'ять великої ємності для зберігання всієї інформації комп'ютера. Якщо при вимкненні комп'ютера інформація в ОП пропадає, тобто вона є енергозалежною, то на інформацію ЗП вимкнення комп'ютера не впливає, тобто це є енергонезалежна пам'ять для довготермінового зберігання інформації.

Процесор виконує обробку інформації та керує роботою інших вузлів комп'ютера.

До складу процесора входять наступні функціональні вузли:

– арифметико-логічний пристрій (АЛП) – набір комбінаційних схем, які виконують арифметичні, логічні та інші операції;

– регістрова пам'ять (РП) – набір програмно доступних регістрів, в яких зберігається найчастіше використовувана в процесорі інформація;

– пристрій керування (ПК) – керує роботою та взаємодією функціональних вузлів комп'ютера.

Лекція 2 (2 год.) Поняття архітектури комп'ютера. Архітектурні принципи Джона фон Неймана.

План лекції:

1. Поняття архітектури комп'ютера.
2. Принципи організації комп'ютера фон Неймана.
3. Неймановські архітектури комп'ютера.

Вперше означення терміну "архітектура комп'ютера" було зроблене в 1964 році розробниками комп'ютера IBM 360 Г. Амдалем та його колегами. Архітектура комп'ютера з їх точки зору – це його структура і поведінка як їх бачить програміст на асемблерній мові. Вона включає наступне: формати даних і команд, методи адресації, систему команд, а також загальну організацію процесора, основної пам'яті і пристроїв введення-виведення. Пізніше А. Пейджез з тієї ж фірми запропонував розуміти під архітектурою комп'ютера інтерфейс між його апаратним та програмним забезпеченням.

Як відомо, в комп'ютері використовується двійкове представлення команд. При написанні програми крім двійкової можуть використовуватись і інші форми представлення команд: вісімкова, шістнадцяткова, символна (мнемонічна). Використання вісімкового і шістнадцяткового кодування дозволяє скоротити записи і спростити роботу програміста. Мнемонічне

кодування спрощує процес написання, читання і відлагодження програми. Основний принцип такого кодування – кожна команда представляється 3-х або 4-х буквеним символом, який показує назву команди. Деякі приклади мнемонічного кодування: *add* – додати, *sub* – відняти, *load* – зчитати дані з пам'яті, *store* – записати дані в пам'ять. Операнди також представляються символічно. Наприклад, команда *add R, Y* означає додавання значення вмісту комірки пам'яті *Y* до вмісту регістра *R*. Зауважимо, що операція виконується над вмістом, а не над адресою комірки пам'яті та регістра. Таким чином, з'являється можливість написання машинних програм в символічній формі. Повний набір символічних назв і правила їх використання утворюють мову програмування, відому як асемблерна мова. Запис деякої програми на асемблерній мові представляє собою символічний запис цієї ж програми, написаної на внутрішній мові комп'ютера, тобто в формі послідовності команд, представлених в двійкових кодах. Тому і з'явилося визначення архітектури комп'ютера як інтерфейсу між його апаратним та програмним забезпеченням.

З наведених вище означень можна зробити висновок про існування множини можливих варіантів архітектури комп'ютера. Розглянемо детальніше складові поняття архітектури з тим, щоб визначити ступінь зміни їх характеристик.

Під даними розуміються числа, представлені в деякій системі числення. В сучасних комп'ютерах дані в більшості випадків представлені в позиційній двійковій системі числення, яка практично витіснила інші способи представлення чисел, наприклад, представлення в позиційній десятковій системі числення, в системі залишкових класів тощо.

В комп'ютерах є три класи даних: вхідні, проміжні та вихідні, які можна охарактеризувати наступною множиною параметрів: кількістю N даних, їх розрядністю n ($n = 1, 2, 3, \dots$), способом кодування M (прямий, обернений чи доповняльний код), формою представлення B (фіксована чи рухома кома), форматом T (місце розміщення знаку, ціле чи дробове число, місце розміщення коми, розрядність порядку, основа порядку, використання знаку порядку чи зміщення, розрядність мантиси, чи використовується нормалізація мантиси). Таким чином, залежно від значення параметрів n , M , B , T можна виділити відповідну архітектуру комп'ютера. Наприклад, це може бути комп'ютер для виконання операцій над 16-розрядними дробовими двійковими даними в форматі з фіксованою комою в доповняльному коді, або це може бути комп'ютер для виконання операцій над 32-розрядними двійковими даними в форматі з рухомою комою.

Команда в комп'ютері зберігається в двійковій формі. Вона вказує тип операції, яка має бути виконаною, адреси операндів, над якими виконується операція, та адреси розміщення результатів виконання операції. Відповідно до цього команда складається з двох частин: коду операції та адресної частини. Залежно від формату команди комп'ютери можна поділити на: комп'ютери з постійним та змінним форматом команди, комп'ютери з одноадресними,

двоадресними та триадресними командами, комп'ютери з вузьким та широким форматом команди (залежно від кількості полів коду операції та адресних полів) і т. д.

Як вказано вище, крім коду операції до складу команди входить адресна частина. Цією частиною визначається місце знаходження даних, над якими виконується операція, задана кодом операції. Даних може бути декілька, і, крім того, вони можуть знаходитися в основній пам'яті, в регістрах процесора чи в запам'ятовуючих елементах інших вузлів комп'ютера. Тому і формати команд в цих випадках будуть різними. Можна здійснити класифікацію архітектури комп'ютера за типом адресованої пам'яті. Залежно від того, який тип пам'яті адресується, розрізняють наступні типи архітектур комп'ютера: стекова, акумуляторна, на основі регістрів загального користування.

При розробці комп'ютера необхідно враховувати багато особливостей вибору системи команд. З одного боку, система команд повинна бути функціонально повною, тобто комп'ютер повинен забезпечувати виконання всіх заданих функцій. З іншого боку, система команд має бути ортогональною, тобто не повинна бути надлишковою. Для нового комп'ютера, який є розширенням відповідної серії, першочерговою є сумісність – програми, які виконуються на одному комп'ютері, повинні виконуватись і на іншому комп'ютері. Є багато рівнів повноти системи команд. Теоретично система команд комп'ютера може включати лише одну команду. Однак програми на базі простих операцій є дуже складними. Існує фундаментальний зв'язок між простотою комп'ютера і складністю програми. В реальних комп'ютерах застосовується система команд, до складу якої входить широкий спектр команд обробки даних, переміщення даних, передачі керування та введення-виведення. За складом системи команд комп'ютери можуть бути поділені на наступні типи:

- комп'ютери з складною (комплексною) системою команд;
- комп'ютери з простою (спрощеною) системою команд;
- комп'ютери з доповненою системою команд;
- комп'ютери з орієнтованою (спеціалізованою) системою команд.

Значна кількість типів архітектури комп'ютера може бути виділена залежно від організації його вузлів, а саме процесора, пам'яті і пристроїв введення-виведення. Наприклад, це можуть бути комп'ютери з паралельною та конвеєрною обробкою даних, з ієрархічною та лінійною пам'яттю і т. д.

Архітектура комп'ютера має визначальний вплив на його споживчі характеристики: коло вирішуваних на комп'ютері задач, продуктивність, ємність основної пам'яті, ємність зовнішньої пам'яті, вартість, організація технічного обслуговування, надійність і т. д.

Для вибору кращої з множини можливих варіантів архітектури комп'ютера потрібно знати зв'язок між архітектурою комп'ютера та його характеристиками, тобто як ті чи інші структурні особливості та організація роботи комп'ютера і його вузлів зв'язані з можливостями, які надаються користувачу, які є альтернативи при створенні комп'ютера і за якими

критеріями повинні прийматися ті чи інші проектні рішення, як зв'язані між собою характеристики окремих пристроїв комп'ютера і який вплив вони мають на загальні його характеристики.

Описана в 1946 році Джоном фон Нейманом архітектура комп'ютера дістала назву його імені. Оскільки це був опис реалізованого Преспером Екертом та Джоном Моучлі універсального комп'ютера ЕКІАС, створеного в Принстонському університеті, часом її іще називають принстонською.

Головні особливості **архітектури комп'ютера Джона фон Неймана**:

1. *Використання двійкової системи числення в обчислювальних машинах.* Перевага перед десятковою системою числення полягає в тому, що пристрої можна робити досить простими, арифметичні і логічні операції в двійковій системі числення також виконуються досить просто.

2. *Програмне управління ЕОМ.* Робота ЕОМ контролюється програмою, що складається з набору команд. Команди виконуються послідовно один за одним. Створенням машини з програмою, яка зберігалася у пам'яті, було покладено початок тому, що ми сьогодні називаємо програмуванням.

3. *Пам'ять комп'ютера використовується не тільки для зберігання даних, але і програм.* При цьому і команди програми і дані кодуються в двійковій системі числення, тобто їх спосіб запису однаковий. Тому в певних ситуаціях над командами можна виконувати ті ж дії, що і над даними.

4. *Комірки пам'яті ЕОМ мають адреси, які послідовно пронумеровані.* У будь-який момент можна звернутися до будь-якої комірки пам'яті на її адресу. Цей принцип відкрив можливість використовувати змінні в програмуванні.

5. *Можливість умовного переходу в процесі виконання програми.* Не дивлячись на те, що команди виконуються послідовно, в програмах можна реалізувати можливість переходу до будь-якої ділянки коду.

В основу роботи цього комп'ютера покладено *принцип програмного керування*. Тобто функціонування цього комп'ютера здійснюється потактово за вказівкою команд програми. Програма разом з даними, які підлягають обробці, спочатку записується до основної пам'яті. В кожному такті, залежно від типу архітектури комп'ютера, виконується команда або частина команди – мікрокоманда. Команда вказує тип виконуваної операції та місце розміщення операндів і результату операції. Для виконання однієї команди необхідно провести наступні дії: записати до програмного лічильника адресу команди, зчитати із основної пам'яті команду за вмістом програмного лічильника, провести дешифрування команди з метою її розпізнання, визначити адреси комірок пам'яті, в яких знаходяться операнди та до яких мають бути записані результати, зчитати ці операнди з пам'яті та подати в арифметико-логічний пристрій для опрацювання, виконати операцію над операндами та записати результати до основної пам'яті. Таким чином проводиться виконання всіх команд програми. По закінченню в основній пам'яті будуть знаходитися результати виконання програми.

Контрольні запитання:

1. Дайте визначення цифрового комп'ютера.
2. Які основні функціональні вузли входять до складу будь-якого комп'ютера?
3. Що входить до поняття архітектури комп'ютера?
4. Назвіть основні типи пам'яті, що входять до складу комп'ютера.
5. Наведіть основні принципи архітектури фон Неймана.
6. На які типи можуть бути поділені комп'ютери за складом системи команд?
7. Яке основна відмінність гарвардської архітектури від фоннеймановської? Які ще ненеімановські архітектури ви знаєте?

Тема 1.2 Типи сучасних комп'ютерів. Основні характеристики комп'ютерів.

Лекція 1 (2 год.) Типи сучасних комп'ютерів.

План лекції:

1. Мікроконтролери – комп'ютери на кристалі.
2. Персональні комп'ютери.
3. Робочі станції, багатотермінальні системи, сервери.
4. Мейнфрейми, суперкомп'ютери.
5. Спеціалізовані комп'ютери.

Залежно від сфери застосування та технічних характеристик, в першу чергу габаритів, продуктивності, ємності пам'яті та ціни розрізняють наступні типи комп'ютерів:

- мікроконтролери – комп'ютери на кристалі, призначені для керування електронними пристроями;
- персональні комп'ютери (personal computers) – комп'ютери, орієнтовані на користування однією особою;
- робочі станції (workstations) – комп'ютери, орієнтовані на вирішення інженерних задач, в першу чергу в складі локальних комп'ютерних мереж;
- багатотермінальні системи – набір стандартних терміналів, підключених до сервера;
- сервери – центральні комп'ютери для побудови інформаційних систем;
- мейнфрейми (mainframes) – великі універсальні комп'ютерні системи;
- кластерні комп'ютерні системи – об'єднання комп'ютерів, що сприймається операційною системою, системним програмним забезпеченням, прикладними програмами і користувачами як єдине ціле;
- суперкомп'ютери – найпотужніші на даний час комп'ютери;
- спеціалізовані комп'ютери. Вони орієнтовані на вирішення задач, котрі неможливо або недоцільно виконувати на універсальних комп'ютерах.

Розглянемо названі комп'ютери детальніше.

Мікроконтролери – комп'ютери на кристалі, призначені для керування електронними пристроями, зокрема побутовими пристроями, виробничими лініями, вимірювальними пристроями і т. д. До складу мікроконтролера входять наступні вузли:

- центральний процесор, розрядністю від 4 до 64 бітів, залежно від потрібної точності обчислень;
- інтерфейси введення-виведення, в першу чергу послідовні порти;
- периферійні пристрої, такі як: таймери та схеми захисту, цифроаналогові та аналогоцифрові перетворювачі;
- пам'ять з довільним доступом для зберігання даних;
- постійна пам'ять типу ROM, EPROM, EEPROM чи Flash для зберігання програми;
- генератор тактів.

Така інтеграція названих пристроїв на кристалі дозволяє забезпечити малі габарити та споживання і сприяє широкому використанню мікроконтролерів у різного роду вбудованих системах. Наприклад, в сучасному автомобілі використовується понад 50 мікроконтролерів. Вони також використовуються в побутовій електроніці, мобільних телефонах, виробничих лініях тощо.

Розробники мікроконтролерів забезпечують спеціальний сервіс для користувачів, зокрема версії, які дозволяють перепрограмування програмної пам'яті ультрафіолетовим світлом, можливість підключення зовнішньої оперативної пам'яті в якості пам'яті програм, та інше. Сучасні мікроконтролери програмуються в коді мови C та мають внутрішні схеми відлагодження.

Персональні комп'ютери (ПК) з'явилися в результаті еволюції мінікомп'ютерів при переході елементної бази з малим і середнім ступенем інтеграції на великі і надвеликі інтегральні схеми. ПК, завдяки низькій вартості, дуже швидко завоювали тверді позиції на комп'ютерному ринку і створили передумови для розробки нових програмних засобів, що орієнтувалися на кінцевого користувача. Це, передусім, дружні по відношенню до користувача інтерфейси, а також проблемно-орієнтовані середовища і інструментальні засоби для автоматизації розробки прикладних програм.

Персональні комп'ютери класифікуються за їх розмірами та конструктивним виконанням наступним чином:

- **Настільні комп'ютери (desktop computers).** До складу настільного комп'ютера входить системний блок (в якому розміщена материнська плата, центральний процесор, основна пам'ять, карта розширення, блок живлення і т. д.), дисплей, клавіатура, мишка. В системний блок також вбудовані драйвер оптичного диску і зовнішня дискова пам'ять. Настільні комп'ютери призначені для офісного використання.

- **Лаптопи чи ноутбуки (laptop or notebooks).** Це близькі за характеристиками до настільного ПК, але конструктивно виконані в придатному для перенесення виконанні.

- **Персональні цифрові асистенти (Personal digital assistants).** Це кишенькові ПК, спеціально створені як персональні асистенти людини. Вони надають наступний сервіс: годинник, комп'ютерні ігри, доступ до мережі Інтернет, електронна пошта, записна книжка, адресна книжка, мобільний телефон, медіа плеєр та інше.

- **Смартфони.** Це мобільні телефони, які мають вбудовану операційну систему та можливості вище описаних персональних цифрових асистентів.

- **Портативні комп'ютери (Portable computers).** Це ПК типу настільних, але виконані в придатному для перенесення та роботи в не офісних умовах конструктивному виконанні.

- **Переносимі комп'ютери (Wearable computers).** Це комп'ютери, які надають інформаційні послуги людині під час її руху в навколишньому

середовищі. До переносимих ПК належать, зокрема, комп'ютери для моніторингу стану людини.

Мінікомп'ютери стали прародичами й іншого напрямку розвитку сучасних 32 та 64-розрядних комп'ютерів, що сьогодні відомі як **робочі станції**. Початкова орієнтація робочих станцій на професійних користувачів (на відміну від ПК, що від початку були орієнтовані на споживача-непрофесіонала) призвела до того, що робочі станції – це добре збалансовані комп'ютерні системи, які, разом з високою продуктивністю, характеризуються великою ємністю основної і зовнішньої пам'яті, мають високошвидкісні внутрішні магістралі, високоякісну і швидкодіючу графічну підсистему і різноманітні пристрої введення-виведення. Ця властивість вигідно відрізняє робочі станції середнього і високого класу від ПК і сьогодні. Навіть найпотужніші ПК не в стані задовольнити зростаючі потреби інженерних задач через наявність в їхній архітектурі ряду вузьких місць.

Багатотермінальні системи

ПК та робочі станції часто застосовуються в якості дорогих дисплеїв і в цьому випадку не повністю використовується їх обчислювальна потужність. Разом з тим, багато користувачів терміналів хотіли б покращити їхні графічні характеристики та мати можливість роботи в багатовіконній системі. Ці проблеми були вирішені шляхом створення багатотермінальних систем, які є набором стандартних терміналів, підключених до сервера. Як тільки стали доступними потужні графічні робочі станції, з'явилася тенденція застосування "підлеглих" терміналів, що використовують робочу станцію в якості локального сервера.

На комп'ютерному ринку багатотермінальні системи займають проміжне положення між персональними комп'ютерами і робочими станціями. Постачальники терміналів заявляють, що їхні вироби ефективніші в вартісному вираженні, ніж робочі станції високого цінового класу, і пропонують збільшений рівень продуктивності у порівнянні з персональними комп'ютерами, що робить цю технологію доступною для широкого кола користувачів. Вартість терміналів складає біля половини вартості близького за конфігурацією ПК без зовнішньої пам'яті і приблизно чверть вартості повністю оснащеної робочої станції.

Типовий термінал включає наступні елементи:

- екран високої роздільної здатності;
- головний процесор, який підтримує двопроцесорну архітектуру;
- окремий графічний співпроцесор, що забезпечує швидше малювання на екрані і прокручування екрану;
- базові системні програми;
- програмне забезпечення сервера;
- локальну пам'ять для дисплею та мережного інтерфейсу, що підтримує протокол TSP/IP та інші мережні протоколи;
- порти для підключення клавіатури і миші.

Термінали відрізняються від ПК і робочих станцій не тільки тим, що не виконують функції звичайної локальної обробки. Робота терміналів залежить від головної системи, до якої вони підключені через мережу. Для того, щоб термінал міг працювати, користувачі повинні встановити програмне забезпечення багатовіконного сервера на головному процесорі, що виконує прикладну задачу. Локальна обчислювальна потужність терміналу зазвичай використовується для виконання програм обробки зображень, а не прикладних програм, які виконуються на головному процесорі. Термінал може відображати на одному і тому ж екрані декілька задач. Користувач може змінювати розміри вікон, їхнє місцезнаходження і маніпулювати ними в будь-якому місці екрана.

Сервери

Прикладні комерційні та бізнесові системи, розраховані на багато користувачів, включаючи системи керування базами даних і обробки транзакцій, великі видавничі системи, мережні системи і системи обслуговування комунікацій, системи розробки програмного забезпечення і обробки зображень, вимагають переходу до моделі обчислень "клієнт-сервер" і розподіленої обробки. В розподіленій моделі "клієнт-сервер" частину роботи виконує сервер, а частину – комп'ютер користувача (в загальному випадку частини сервера і користувача можуть виконуватись і на одному комп'ютері). Існує декілька типів серверів для різних застосувань: файловий сервер, сервер бази даних, принт-сервер, обчислювальний сервер, сервер застосувань. Таким чином, тип сервера визначається ресурсом, яким він володіє (файлова система, база даних, принтери, процесори або прикладні пакети програм).

З іншого боку існує класифікація серверів за масштабом мережі, в якій вони використовуються: сервер робочої групи, сервер відділу або сервер підприємства (корпоративний сервер). Ця класифікація надто умовна. Наприклад, розмір групи може змінюватися в діапазоні від декількох людей до декількох сотень людей, а сервер відділу може обслуговувати від 20 до 150 користувачів. Очевидно, залежно від числа користувачів і характеру вирішуваних ними завдань, вимоги до складу обладнання і програмного забезпечення сервера, до його надійності та продуктивності суттєво відрізняються.

Файлові сервери невеликих робочих груп (не більше 20-30 людей) простіше всього реалізуються на платформі персональних комп'ютерів і програмному забезпеченні *Novell NetWare*. Файл-сервер в даному випадку виконує роль центрального сховища даних. Типовими до складу невеликих файл-серверів входять процесор, основна та зовнішня пам'ять, а також адаптер *Ethernet*. До складу таких серверів часто включаються дисковод компакт-дисків. Графіка для більшості серверів несуттєва, тому достатньо мати звичайний монохромний монітор з невисокою роздільною здатністю. Бажано застосувати пристрій безперебійного живлення.

Для файл-серверів загального доступу, з якими водночас можуть працювати декілька десятків, а то і сотень людей, простої однопроцесорної

платформи і програмного забезпечення *Novell* може виявитися недостатньо. В цьому випадку використовуються потужні багатопроцесорні сервери з можливостями нарощування основної пам'яті та дискового простору, швидкими інтерфейсами дискового обміну (типу Fast SCSI-2, Fast&Wide SCSI-2 і Fiber Channel) і декількома мережними інтерфейсами. Ці сервери використовують операційну систему UNIX, мережні протоколи TCP/IP і NFS. На базі багатопроцесорних UNIX-серверів зазвичай будуються також сервери баз даних великих інформаційних систем, бо на них покладається основне навантаження з обробки інформаційних запитів. Сервери подібного типу отримали назву суперсерверів.

За загальносистемною продуктивністю, функціональними можливостями окремих компонентів, стійкістю до відмов, а також ступенем підтримки багатопроцесорної обробки, системного адміністрування і дискових масивів великої ємності суперсервери вийшли в нинішній час на один рівень з мейнфреймами. Сучасні суперсервери характеризуються наявністю двох або більше центральних процесорів, багатоштинною структурою, мають достатні можливості нарощування дискового простору і обчислювальної потужності, засоби забезпечення надійності зберігання даних і захисту від несанкціонованого доступу.

Великі універсальні комп'ютерні системи (мейнфрейми) і до сьогоднішнього дня залишаються найпотужнішими (не рахуючи суперкомп'ютерів) комп'ютерними системами загального призначення, що забезпечують безперервний цілодобовий режим експлуатації. Вони можуть включати один або декілька процесорів, кожний з яких, у свою чергу, може споряджатися векторними спеціалізованими процесорами (прискорювачами операцій з продуктивністю суперкомп'ютерів). Зазвичай мейнфрейми асоціюються з великими за габаритами комп'ютерами, що вимагають спеціально обладнаних приміщень із системами водяного охолодження і кондиціонування. Однак прогрес в області елементної та конструкторської бази дозволив істотно зменшити габарити основних пристроїв. Поряд з надпотужними мейнфреймами, що вимагають організації двоконтурної водяної системи охолодження, є менш потужні моделі, для охолодження яких достатньо примусової повітряної вентиляції, та моделі, побудовані за блоково-модульним принципом, які не вимагають спеціальних приміщень і кондиціонерів.

Основними постачальниками мейнфреймів є відомі комп'ютерні компанії *IBM*, *Amdahl* (в даний час в складі концерну *Fujitsu*), *ICL*, *Fujitsu Siemens Computers*, *Wincor Nixdorf AG*, *Siemens Business Services* і деякі інші, але провідна роль належить безумовно компанії *IBM*. Саме архітектура системи *IBM/360*, випущеної в 1964 році, і її наступні покоління стали зразком для наслідування. Так, в СРСР протягом багатьох років випускалися машини ряду ЕС ЕОМ, що є аналогом цієї системи. В архітектурному плані мейнфрейми є багатопроцесорними системами, що містять один або декілька центральних і периферійних процесорів зі спільною пам'яттю, зв'язаних між

собою високошвидкісними магістралями передачі даних. При цьому основне обчислювальне навантаження покладено на центральні процесори, а периферійні процесори (в термінології IBM – селекторні, блок-мультиплексні, мультиплексні канали і процесори телеопрацювання) забезпечують роботу з широкою номенклатурою периферійних пристроїв.

Стрімкий ріст продуктивності персональних комп'ютерів, робочих станцій і серверів призвів до зниження потреби в мейнфреймх. Однак цей процес останнім часом дещо уповільнився. Основною причиною відродження інтересу до мейнфреймів експерти вважають складність переходу до розподіленої архітектури клієнт-сервер, що виявилася вищою, ніж припускалося. Крім того, багато користувачів вважають, що розподілене середовище, на противагу мейнфреймам, не є достатньо надійним для найвідповідальніших застосувань.

Головним недоліком мейнфреймів у нинішній час залишається відносно низьке співвідношення продуктивність/вартість. Однак постачальники мейнфреймів докладають значних зусиль для поліпшення цього показника. Слід також пам'ятати, що в світі існує величезна інстальована база мейнфреймів, на якій працюють десятки тисяч прикладних програмних систем. Відмовитися від роками напрацьованого програмного забезпечення просто нерозумно. Тому в нинішній час очікується зростання об'ємів продажу мейнфреймів, які з одного боку, дозволять модернізувати існуючі системи, забезпечивши скорочення експлуатаційних видатків, а з іншого боку, створять нову базу для найвідповідальніших застосувань.

Кластерні комп'ютерні системи

Двома основними проблемами побудови комп'ютерних систем для критично важливих застосувань, зв'язаних з обробкою транзакцій, керуванням базами даних і обслуговуванням телекомунікацій, є забезпечення високої продуктивності та тривалого функціонування систем. Найефективнішим засобом для досягнення заданого рівня продуктивності є застосування паралельних архітектур, які піддаються масштабуванню. Завдання забезпечення тривалого функціонування системи має три складових: надійність, готовність і вартість обслуговування. Всі три складові передбачають, в першу чергу, боротьбу з несправностями системи, що породжуються відмовами і збоями в її роботі. Ця боротьба ведеться по всіх трьох напрямках, що взаємозв'язані і застосовуються спільно.

Робота будь-якої кластерної системи визначається двома головними компонентами: високошвидкісним механізмом зв'язку процесорів між собою і системним програмним забезпеченням, що надає клієнтам прозорий доступ до системного сервісу.

В даний час широке розповсюдження отримала технологія паралельних баз даних. Ця технологія дозволяє великій кількості процесорів поділяти доступ до єдиної бази даних. Розподіл завдань між процесорними ресурсами і паралельне їх виконання дозволяє досягнути вищого рівня пропускної спроможності транзакцій, підтримувати більше число одночасно працюючих

користувачів і прискорити виконання складних запитів. Для вирішення цих завдань використовується архітектура зі спільними (розподіленими) дисками. Це типовий випадок побудови кластерної системи. Ця архітектура підтримує єдину базу даних при роботі з декількома комп'ютерами, об'єднаними в кластер (зазвичай такі комп'ютери називаються вузлами кластера), кожний з яких працює під керуванням своєї копії операційної системи. В таких системах всі вузли поділяють доступ до загальних дисків, на яких власне і розміщується єдина база даних. Продуктивність таких систем може збільшуватися як шляхом нарощування числа процесорів і ємності основної пам'яті в кожному вузлі кластера, так і шляхом збільшення кількості самих вузлів. У випадку відмови одного з таких вузлів, вузли, що залишилися, можуть взяти на себе завдання, що виконувалися на вузлі, який відмовив, не зупиняючи загальний процес роботи з базою даних. Оскільки логічно в кожному вузлі системи є образ бази даних, доступ до неї буде забезпечуватися до тих пір, доки в системі є принаймні один справний вузол.

До класу **суперкомп'ютерів** належать комп'ютери, що мають максимальну в даний час продуктивність, а також максимальну ємність основної та зовнішньої пам'яті. Вони асоціюються з великими розмірами, великими завданнями, гранично високими характеристиками. Швидкий розвиток комп'ютерної індустрії призводить до відносності даного поняття. Суперкомп'ютер десятирічної давності сьогодні під це визначення вже не потрапляє. Наприклад, продуктивність персональних комп'ютерів, що використовують *Pentium-II/300MHz*, є близькою до продуктивності суперкомп'ютерів середини 70-х років, проте за сьогоднішніми мірками суперкомп'ютерами не є ні ті, ні інші.

От лише невеличкий список областей людської діяльності, де необхідно використовувати суперкомп'ютери: автомобілебудування; нафто- і газовидобуток; фармакологія; прогноз погоди і моделювання зміни клімату; сейсмозвідка; проектування електронних пристроїв; синтез нових матеріалів, генні дослідження.

Titan – Cray XK7

Местоположение: США. Производительность: 17,59 петафлопс. Теоретический максимум производительности: 27,11 петафлопс. Мощность: 8,2 МВт. Наиболее производительный из когда-либо созданных на Западе суперкомпьютеров, а также самый мощный компьютерный кластер под маркой компании *Cray* находится в США в Национальной лаборатории Оук-Ридж. Несмотря на то, что находящийся в распоряжении американского Министерства энергетики суперкомпьютер официально доступен для любых научных исследований, в октябре 2012 года, когда *Titan* был запущен, количество заявок превысило всякие пределы.

Tianhe-2 / Млечный путь-2

Местоположение: Китай. Производительность: 33,86 петафлопс. Теоретический максимум производительности: 54,9 петафлопс. Мощность: 17,6 МВт. С момента своего первого запуска «Тяньхэ-2», или «Млечный-путь-

2», вот уже около двух лет является лидером Top-500. Этот монстр почти в два раза превосходит по производительности №2 в рейтинге – суперкомпьютер TITAN. Разработанный Оборонным научно-техническим университетом Народно-освободительной армии КНР и компанией Inspur «Тяньхэ-2» состоит из 16 тысяч узлов с общим количеством ядер в 3,12 миллиона. Оперативная память всей это колоссальной конструкции, занимающей 720 квадратных метров, составляет 1,4 петабайт, а запоминающего устройства – 12,4 петабайт.

За допомогою універсальних комп'ютерів та комп'ютерних систем (УКС) можна вирішувати багато задач наукового, виробничо-технічного та іншого характеру. Однак існують надзвичайно важливі класи задач і окремі задачі, для розв'язку яких УКС недостатньо. Загальний аналіз причин створення і використання **спеціалізованих комп'ютерних систем (СКС)** показує, що ці причини можна віднести до трьох основних груп.

Перша група об'єднує причини, що виникли внаслідок суперечностей між формальними математичними методами постановки і розв'язку задач, з одного боку, і загальними принципами організації та функціонування, а також технічними можливостями УКС, з іншого боку. Саме математична сутність задач часто обумовлює необхідність створення СКС для їх розв'язку. Як приклади тут можна навести нові нестандартні та неалгоритмічні методи, системи алгебраїчних, диференціальних та інтегральних рівнянь великої розмірності, логічні та імовірісно-статистичні задачі, дії над матрицями та векторами, задачі в багатовимірних просторах та багато інших.

До другої групи входять причини, які обумовлені змістовною стороною задач, вирішуваних СКС, та відображають специфіку відповідних предметних областей.

Третя група причин обумовлена особливими вимогами до якості реалізації комп'ютерних систем, які зазвичай полягають в екстремалізації (тобто в максимальному наближенні до теоретичних границь) деяких їх характеристик, наприклад, продуктивності, надійності (безвідмовності, живучості, відновлюваності, довговічності та ін.), вартості, точності, габаритів, маси і т.п. Сюди ж належать вимоги, що визначають такі якості комп'ютерних систем, як їх повна або часткова імплантація (конструктивне та функціональне суміщення) в інші системи, інформаційне поєднання з ними, пристосованість до умов експлуатації та кваліфікації обслуговуючого персоналу і т.д.

Друга особливість пов'язана з тим, що реальні СКС є складними програмно-технічними комплексами, в яких на інженерному рівні необхідно задовольнити багато суперечливих вимог. Тому досягнення оптимальних і функціональних якостей СКС може бути проблематичним і доцільніше визначати ці якості як оптимізовані, тобто такі, що тією чи іншою мірою наближаються до оптимальних. Аналіз математичних методів оптимізації СКС показує, що вони дозволяють, певною мірою, виявляти недоліки таких систем, їхні "слабкі місця", простежити взаємозв'язок характеристик системи, визначити загальний напрямок підвищення їх ефективності та оцінити різні

варіанти СКС. Однак ці методи не дають ніяких конструктивних рішень і шляхів удосконалення СКС, не визначають змістовної сторони різних варіантів їх організації та реалізації. Генезис таких варіантів формальними математичними методами неможливий. Тому процес створення оптимізованих СКС має характер багатоступеневої ітераційної процедури, де в різних відношеннях комбінуються формальні та конкретно-змістовні методи, що відіграють аналітичну (оціночну) та синтетичну (генеративну) ролі.

Таким чином, СКС – це комп'ютерні системи для розв'язку великого числа відносно вузьких класів задач, оптимізовані в певній критеріальній сукупності.

Для СКС характерні наступні риси, які відрізняють їх від універсальних комп'ютерних систем:

- орієнтація структури на вирішувани задачі;
- вузький, в основному постійний клас вирішуваних задач;
- особливі вимоги до точності, часто нестандартна довжина розрядної сітки;
- спеціальна система обміну, в тому числі наявність аналого-цифрових та цифро-аналогових каналів зв'язку;
- використання орієнтованих на область застосування мов програмування та широкі можливості їх апаратної інтерпретації;
- наявність спеціальних функцій і процедур в наборі операцій та команд;
- необхідність обробки вхідних даних в темпі їх поступлення та видачі результатів обчислень в темпі поступлення вхідних даних;
- суміщення в часі приймання, обробки та видачі даних;
- висока продуктивність;
- малі габарити;
- низька споживана потужність;
- орієнтація конструкції на конкретне застосування.

Лекція 2 (2 год.) Основні характеристики комп'ютерів.

План лекції:

1. Продуктивність як основна характеристика комп'ютера.
2. Вплив архітектури комп'ютера на його продуктивність.

Продуктивність є однією з основних характеристик комп'ютера, яка залежить як від технологічних, так і від архітектурних рішень. В процесі розвитку комп'ютерної техніки з'явилося декілька методик вимірювання продуктивності. Вони дозволяють розробникам і користувачам здійснювати вибір між альтернативами на основі кількісних показників.

Припустимо, що використано два комп'ютери для виконання тієї ж програми. Якщо перший комп'ютер виконав програму за менший час у порівнянні з другим, можна говорити, що перший комп'ютер є швидшим. *Час виконання програми* включає час роботи процесора, час звернення до дискової

пам'яті, час звернення до основної пам'яті, час введення-виведення даних і накладні витрати операційної системи. Оскільки при роботі в мультипрогравному режимі під час очікування введення-виведення для однієї програми, процесор може виконувати іншу програму, то система не обов'язково мінімізуватиме час виконання даної конкретної програми. Тому вказаний підхід до порівняння комп'ютерів не є досконалим.

Іншим підходом до порівняння комп'ютерів є порівняння *часу роботи процесора*, необхідного для виконання заданої програми, який не включає час очікування введення-виведення або час виконання іншої програми. Час роботи процесора може бути виражений кількістю тактів синхронізації для даної програми, помноженою на тривалість такту синхронізації.

Важливою та часто вживаною характеристикою є *середня кількість тактів синхронізації процесора на одну команду CPI* (clock cycles per instruction). При відомій кількості виконуваних команд в програмі ця характеристика дозволяє швидко оцінити час роботи процесора, необхідний для виконання заданої програми.

Досконалішою одиницею, яку можна використати для порівняння комп'ютерів, є *продуктивність*, тобто загальна кількість обчислювальної роботи, яку комп'ютер виконує за фіксований часовий інтервал. Якщо час виконання деякої програми позначити через T , то продуктивність P комп'ютера можна визначити наступним чином: $P = 1/T$. Тоді порівняння двох комп'ютерів X і Y можна виконати за наступними правилами: якщо $1/T_x > 1/T_y$ тобто $T_y > T_x$, то комп'ютер X є швидшим. В сучасних комп'ютерах продуктивність вимірюється в мільйонах операцій за секунду – MIPS. Таким чином, продуктивність може бути визначена як зворотна до часу виконання величина, причому швидші комп'ютери при цьому матимуть вищий рейтинг кількості операцій за одиницю часу.

Позитивними сторонами кількості операцій за одиницю часу як одиниці оцінки продуктивності комп'ютера є те, що цю характеристику легко зрозуміти, особливо покупцю, і що швидший комп'ютер характеризується більшим числом операцій за одиницю часу. Проте використання цієї одиниці як метрики для порівняння натрапляє на дві проблеми. По-перше, вона залежить від набору команд процесора, що ускладнює порівняння комп'ютерів з різними системами команд. По-друге, навіть на одному і тому ж комп'ютері вона змінюється від програми до програми.

Вимірювання продуктивності комп'ютерів при вирішенні науково-технічних задач, в яких переважно використовується представлення даних в форматі з рухомою комою, завжди викликало особливий інтерес. Саме для таких обчислень вперше постало питання про вимірювання продуктивності, а за досягнутими показниками часто робилися висновки про загальний рівень розробок комп'ютерів. Зазвичай для *науково-технічних завдань продуктивність комп'ютера* оцінюється в кількості операцій з рухомою комою за секунду FLOPS (Floating Point Operations Per Second). В сьогоденні комп'ютерах це мільйони та мільярди операцій з рухомою комою за секунду -

MFLOPS, GFLOPS.

Потрібно відзначити, що вищеназвані одиниці вимірювання - такт (або частота) синхронізації, середня кількість тактів на команду і продуктивність комп'ютера є взаємозв'язаними. Неможливо змінити жодну з них ізольовано від іншої, оскільки базові технології, використовувані для зміни кожної з цих характеристик, взаємозв'язані: частота синхронізації визначається технологією виготовлення апаратних засобів і функціональною організацією процесора; середня кількість тактів на команду залежить від функціональної організації і архітектури системи команд; а кількість виконуваних в програмі команд визначається архітектурою системи команд і технологією компіляторів. Коли порівнюються два комп'ютери, необхідно розглядати всі три компоненти, щоб зрозуміти відносну продуктивність.

Контрольні запитання:

1. Назвіть типи сучасних комп'ютерів.
2. Які вузли входять до складу мікроконтролерів?
3. Назвіть види персональних комп'ютерів?
4. Яке призначення спеціалізованих комп'ютерних систем?
5. Назвіть основні характеристики комп'ютерів.

Тема 1.3 Форм-фактори материнської плати. Компоненти системної плати.

Лекція 1 (2 год.) Форм-фактори материнської плати. Структура материнської плати. Системна логіка.

План лекції:

1. Поняття форма-фактора материнської плати.
2. Види форма-факторів.
3. Мостова структура материнської плати. Чип-сет.

Форм-фактор, або типорозмір системної плати, це стандарт, який визначає її габарити, параметри електроживлення, розташування монтажних елементів, розміщення різних компонентів (розташування на ній інтерфейсів шин, портів вводу/виводу, процесорного гнізда і слотів для оперативної пам'яті, а також тип роз'єму для підключення блоку живлення). Специфікації форм-фактора також містять вимоги до електричних і механічних параметрів блоку живлення і корпусу. У останніх версіях форм-фактора визначаються і вимоги до системи охолодження комп'ютера.

Перша материнська плата з розробленим форм-фактором з'явилася до 12 серпня 1981 року у комп'ютері специфікації PC. У 1983 році в IBM розробили нову специфікацію - XT PC. На сьогоднішній день існує чотири переважаючих типорозміри материнських плат - AT, ATX, LPX і NLX. Крім того, є зменшені варіанти формату AT (Baby-AT), ATX (Mini-ATX, microATX) і NLX (microNLX). Більш того, недавно випущене розширення до специфікації microATX, що додає до цього списку новий форма-фактор - FlexATX.

Форм-фактор AT поділяється на дві різні за розмірами модифікації – AT і Baby AT. Повнорозмірна материнська плата цього форм-фактора була розроблена для комп'ютерів XT PC. Містилася вона в корпусі Tower або Desktop. Через рік розміри були зменшені і з'явився новий форм-фактор - Baby-AT. До 1996 року материнські плати Baby-AT були, мабуть, найпоширенішими. Усі AT плати мають спільні риси – послідовні і паралельні порти, що приєднуються до материнської плати через сполучні планки. Вони також мають один роз'єм клавіатури в задній частині. Гніздо під процесор встановлюється на передній стороні плати. Сьогодні цей формат сходить зі сцени.

Застарілі формати: Baby-AT; повнорозмірна плата AT; LPX.

Сучасні формати: ATX; Mini-ATX; MicroATX.

Впроваджені Формати: Mini-ITX і Nano-ITX; Pico-ITX; FlexATX; NLX; WTX, SEB; BTX, MicroBTX і PicoBTX.

Форм-фактор LPX (mini-LPX).

У 1987 році компанія Western Digital розробила системну плату з новим форм-фактором LPX. Призначався для використання в корпусах Slimline чи Low-Pro file. Плати розширення розміщені на вертикальній стійці паралельно

материнській платі. Стійка підключається до плати. Ще одне нововведення – це інтегрований на материнську плату відеочип.

Форм-фактор ATX (mini-ATX, micro-ATX, Flex-ATX).

В ATX втілилися кращі сторони і Baby AT і LPX. В результаті вийшло:

- інтегровані роз'єми портів вводу-виводу;
- більш зручний доступ до модулів пам'яті;
- зменшена відстань між платою і дисками;
- рознесення процесора і слотів для плат розширення;
- покращена взаємодія з блоком живлення;
- напруга 3.3 В.

Форм-фактор ATX в усіх його модифікаціях стає є найбільш популярним.

Форм-фактор NLX (Mini-NLX).

У 1997 році, як розвиток ідеї LPX з'явилася специфікація форм-фактора NLX. Формат націлений на застосування в низькопрофільних корпусах. Плати NLX схожі з платами LPX, але вони розраховані на системи з новими процесорами. До того ж вони обладнані портом AGP. Внаслідок нових вимог до охолодження, елементи плати розміщені таким чином, щоб якомога менше заважати циркуляції повітря в корпусі. Змінилося кріплення самої плати. Порти вводу/виводу зміщені до краю плати.

Основні риси материнської плати NLX:

- стійка для карт розширення, що знаходиться на правому краї плати;
- процесор розташований у лівому передньому куті плати, прямо напроти вентилятора;
- розміщення високих компонентів у лівому кінці плати для можливості розміщення на стійці повнорозмірних карт розширення;
- розміщення на задньому кінці плати блоків роз'ємів вводу/виводу одинарної і подвійної висоти, для розміщення максимальної кількості конекторів.

Форм-фактор ВТХ.

Офіційне представлення специфікації The Balanced Technology Extended (ВТХ) 1.0 Public Release відбулося в липні 2004 р. Типоразмер ВТХ розроблений з метою уніфікації інтерфейсів для настільних обчислювальних систем і поліпшення умов функціонування компонентів по електричних, механічних і термічних параметрах. Специфікації ВТХ описують механічні і електричні інтерфейси системних плат, шасі, блоків живлення і інших системних компонентів.

Головні переваги форм-фактора ВТХ перед АТХ такі:

- можливість вживання низькопрофільних компонентів для збірки малогабаритних платформ;
- розміщення елементів системи усередині корпусу з напрямків потоків повітря і термічного балансу;
- масштабованість в рамках доступних модифікацій;
- використання малогабаритних блоків живлення;

- оптимальна конструкція кріплення системної плати, якісні механічні елементи для установки масивних компонентів.

Персональний комп'ютер складається з деякої кількості пристроїв, які так чи інакше підключені до материнської плати і займаються тим, що приймають, оброблюють і передають певну інформацію. Логічною організацією цієї роботи займаються чіпсети.

Чіпсет (Chip Set, Chipset, чіп) – набір мікросхем системної логіки, який здійснює взаємодію елементів системи один з одним і зовнішніми пристроями. Модель материнської плати є однією з основних характеристик системної плати, яка багато в чому визначається чіпсетом.

Фізично чіпсет – це одна або декілька мікросхем, спеціально розроблених для "обов'язків" мікропроцесора, на які покладається основне навантаження по забезпеченню центрального процесора даними і командами, а також, по управлінню периферією (відео карти, звукова система, оперативна пам'ять, дискові накопичувачі і різні порти вводу/виводу).

У загальному випадку саме чіпсет обумовлює не тільки характеристики і продуктивність материнської плати, але і забезпечує підтримку периферійного обладнання різних стандартів.

Для користувачів, вже знайомих з архітектурою ПК, можна додати, що чіпсет материнської плати крім іншого виконує функції таких елементів комп'ютера, як контролер переривань, контролер енергонезалежній пам'яті (BIOS), системний таймер, контролер клавіатури і миші, контролер кеш-пам'яті, контролер дискових накопичувачів і т. д.

Зазвичай в одну з мікросхем набору входять також годинник реального часу зі CMOS-пам'яттю і іноді - клавіатурний контролером, однак ці блоки можуть бути присутніми і у вигляді окремих чіпів. В останніх розробках до складу мікросхем наборів для інтегрованих плат стали включатися і контролером зовнішніх пристроїв. Серед іншого чіпсет визначає:

- тип і швидкодія процесора, який можна підключити до материнської плати;
- тип і максимально допустимий обсяг оперативної пам'яті;
- тип і кількість пристроїв PCI і AGP, які можуть працювати з даним комп'ютером;
- тип і кількість пристроїв, що підключаються до шин SCSI / ISA (жорсткі диски, приводи CD-ROM, DVD і т. д.);
- моделі підключається до комп'ютера клавіатури і миші (USB, PS / 2);
- тип підтримуваних платою портів комп'ютера.

На друкованій платі розташовуються всі компоненти материнської плати і роз'єми для підключення плат розширення і периферійних пристроїв. Нижче на рисунку (5.2) відображена структурна схема розташування компонентів на друкованій платі.

В залежності від типу процесора та материнської плати контролер пам'яті може бути вбудованим в північний міст або процесор, в даному прикладі контролер розташований в процесорі.

Іноді контролер шини PCI-express вбудовують в процесор поряд з контролером пам'яті. В даному прикладі необхідність у північному місті відпадає і чіпсет виконують на основі одної інтегральної схеми, які відповідає за взаємодію з платами розширення та периферійними пристроями.

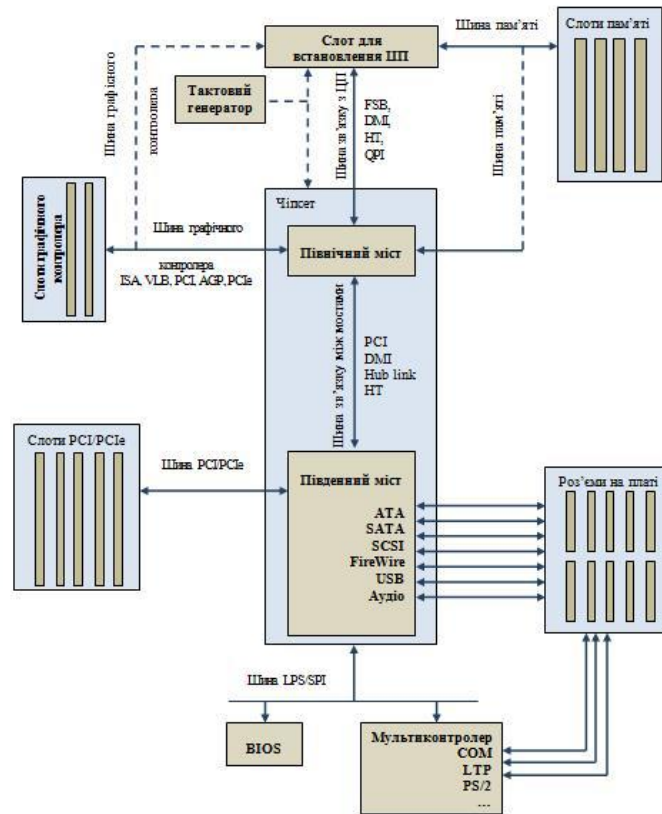


Рисунок 1.1 – Схема розташування компонентів на друкованій платі.

Основні характеристики сучасних материнських плат, на які слід звертати увагу при покупці або модернізації комп'ютера:

- компанія-виробник;
- форм-фактор
- тип встановленого на платі чіпсета;
- тип і швидкодія підтримуваних платою процесорів;
- тип і швидкодія підтримуваних платою модулів оперативної пам'яті;
- можливість розширення;
- швидкість;
- система охолодження;
- стабілізація;
- електроживлення;
- наявність вбудованої графіки;
- наявність вбудованого звуку;
- можливість розгону;

- комплектація;
- зовнішній вигляд;
- OEM або Retail упаковка. RETAIL - це плата, упакована з метою виключення механічних пошкоджень в спеціально сконструйовану коробку (картонні або пінопластові «вставки» складної форми, прокладки, коробки всередині коробки і т.д.). OEM - це плати, офіційно призначені не для продажу в роздрібних торгових мережах, а для використання складальниками готових комп'ютерів;
- кількість роз'ємів для модулів пам'яті;
- зручність збірки;
- Тип НМСЛ в основному визначає функціональні можливості плати. Набір системної логіки складається з двох мікросхем (ще говорять: має дворівневу архітектуру):

* **North Bridge** (Північний міст) - здійснює взаємодію процесора, пам'яті і графічної підсистеми. Містить кеш, контролери оперативної пам'яті, інтерфейс між шиною процесора і PCI, AGP. Все це реалізовано на одному кристалі. Частота роботи цієї мікросхеми дорівнює тактовій частоті материнської плати. Сучасні North Bridge працюють на високих тактових частотах і тому додатково обладнані пристроями охолодження.

* **South Bridge** (Південний міст) - більш повільна мікросхема, інтегрує в систему жорсткі диски, шини PCI, USB, послідовний і паралельний порти і т.п.. Цей компонент відповідає за роботу шини ISA (в наявності є контролер прямого доступу і контролер переривань цієї шини), контролерів IDE і USB, а також реалізує функції пам'яті CMOS, годин і т. д. Ця мікросхема містить велику кількість буферної пам'яті для прискорення обміну з швидкодіючою частиною. Один і той же тип мікросхеми South Bridge може використовуватися, як правило, в декількох наборах системної логіки, тобто може працювати з декількома типами North Bridge.

Лекція 2 (2 год.) Інтерфейси. Особливості реалізації інтерфейсів.

План лекції:

1. Поняття інтерфейсу.
2. Види інтерфейсів.
3. Інтерфейси сучасних комп'ютерів.

Тлумачний словник по обчислювальним системам 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 забезпечує 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 наступні основні класи:

- системні інтерфейси ЕОМ;
- периферійного устаткування (загальні й спеціалізовані);
- програмно-керованих модульних систем і приладів;
- інтерфейси мереж передачі даних і інше.

До інтерфейсу ПК можна віднести порти, слоти, роз'єми, шини.

Головне завдання інтерфейсу - передача даних і керування цим процесом. Тому найважливішою його характеристикою є пропускна здатність.

Продуктивність інтерфейсу виражається в обсязі даних, переданих за одиницю часу. Вона залежить від розрядності, тобто числа бітів інформації, що одночасно проходять по кабелю або шлейфу, і робочої частоти, з якою відбувається передача даних.

Контролер (адаптер, карта розширення, плати) - електронна схема, що управляє зовнішнім пристроєм ПК. Плата контролера дозволяє материнським платам звертатися до спеціалізованих жорстким дискам, а також до сканерів. Проте в даний час більшість жорстких дисків підключається безпосередньо до материнської плати і в картах контролерів не потребує.

За зовнішнім виглядом карти нагадують мініатюрні материнські плати, що може стати причиною виникнення різних проблем. Тому інженерам довелося створювати карти різних розмірів, а також різних розмірів слоти, які їм відповідають. Тепер встановлюючи карту певного розміру в підходящий для неї слот, можна точно бути упевненим, що саме цей слот призначений для даної карти.

Плати адаптерів обумовлюють подальшу спеціалізацію комп'ютера, додаючи нові функціональні можливості. Додавання нового адаптера означає додавання відповідного пристрою. Карти розширення мають загальне призначення, за невеликим винятком можна підключити будь-яку плату розширення в будь слот.

Слот – уніфікований роз'єм на материнській платі для підключення плат розширення. Через такий роз'єм контролери підключаються безпосередньо до системної магістралі передачі даних у комп'ютері-шині. Деякі контролери можуть керувати відразу декількома пристроями. Слоти розширення різних карт розташовані в одному ряду, один біля одного, інші слоти, розташовані розташованих на материнській платі не є слотами розширення. Протягом багатьох років в комп'ютерах використовувалися слоти десятка різних розмірів. У процесі природного відбору їх число скоротилося.

Шина – група електричних з'єднань з'єднує кілька компонентів в цифровій системі. Сучасна системна шина - це не просто велика кількість мідних провідників, розташованих поруч і з'єднують окремі пристрої. Це, перш за все, протокол, за допомогою якого відбувається обмін даними

Порт - електронний блок, за допомогою якого комп'ютер обмінюється даними з іншими пристроями. Комп'ютери мають безліч портів, які призначені для приєднання до них різних кабелів через їх роз'єми. З функціональної точки

зору порти є стандартними, а з фізичної - розташування портів варіюється. Контактні роз'єми більшості портів розташовані на системній платі, деякі порти використовують плату розширення.

Асинхронний послідовний інтерфейс - це основний тип інтерфейсу, за допомогою якого здійснюється взаємодія між комп'ютерами. Термін асинхронний означає, що при передачі даних не використовуються жодні синхронізуючі сигнали і окремі символи можуть передаватися з довільними інтервалами.

Кожному символу, який передається через послідовне з'єднання, повинен передувати стандартний стартовий сигнал, а завершувати його передачу повинен стоповий сигнал. **Стартовий сигнал** – це нульовий біт, названий стартовим бітом. Його призначення – повідомити приймаючий пристрій про те, що наступні вісім бітів є байтом даних. Після символу передаються один або два стопових біта, передачі символу, що сигналізують про закінчення.

У приймаючому пристрої символи розпізнаються по появі стартових і стопових сигналів, а не по моменту їх передачі. Асинхронний інтерфейс орієнтований на передачу символів (байтів), а при передачі використовується приблизно 20% інформації лише для ідентифікації кожного символу. Термін послідовний означає, що передача даних відбувається по одиночному провідникові, а біти при цьому передаються послідовно, один за іншим.

Такий тип зв'язку характерний для телефонної мережі, в якій кожен напрям обслуговує один провідник. До послідовних портів можна підключити: модеми, плотери, принтери, інші комп'ютери, пристрої прочитування штрих-кодів або схему управління пристроями.

В паралельних портах для одночасної передачі байта інформації використовується вісім ліній. Цей інтерфейс відрізняється високою швидкістю, часто застосовується для підключення до комп'ютера принтера, а також для з'єднання комп'ютерів (при цьому вище швидкість передачі даних, чим при з'єднанні через послідовні порти: 4, а не 1 біт за раз).

До паралельних портів може підключатися все: від накопичувачів на магнітній стрічці до CD-ROM. Часто двонаправлений паралельний порт використовується для передачі даних між двома комп'ютерами, наприклад між настільним і портативним. Для зв'язку двох комп'ютерів через паралельний порт потрібний спеціальний кабель. У Windows 9x включена спеціальна програма, яка називається Пряме кабельне з'єднання (Direct Cable Connection), яка дозволяє з'єднати два комп'ютери через модемний нуль-кабель.

Недоліки:

- знижена перешкодозахищеність
- зменшена максимально допустима довжина кабелю
- незручність при складанні
- підвищена складність в інтерфейс мікросхеми.

- необхідність забезпечення синхронізації переданих електричних сигналів як на кінцях кабелю, так і на його окремих провідниках, що з урахуванням високої тактової частоти і перехресних наведень є непростим завданням.

В даний час для настільних і портативних комп'ютерів розроблено два високошвидкісні пристрої з послідовною шиною, що отримали назву **USB** (Universal Serial Bus - універсальна послідовна шина) і **IEEE 1394**, названа також i.Link або Fireware. Можливості цих високошвидкісних комунікаційних портів набагато вищі за стандартні паралельні і послідовні порти, які встановлені в більшості сучасних комп'ютерах.

Перевага нових портів полягає в тому, що їх можна використовувати як альтернативу SCSI для високошвидкісних з'єднань з периферійними пристроями, і в тому, що до них можуть під'єднуватися всі типи зовнішніх периферійних пристроїв.

USB (Universal Serial Bus – універсальна послідовна шина) є промисловим стандартом розширення архітектури PC, орієнтованим на інтеграцію з телефонією і пристроями побутової електроніки. Архітектура USB визначається такими критеріями:

1. Легко реалізоване розширення периферії PC.
2. Дешеве рішення, підтримує швидкість передачі до 480 Мбіт / с.
3. Повна підтримка в реальному часі передачі аудіо та стислих відео даних.
4. Гнучкість протоколу для змішаної передачі ізоморфних даних і асинхронних повідомлень.
5. Інтеграція в технологію пристроїв, що випускаються.
6. Доступність в PC всіх конфігурацій і розмірів.
7. Відкриття нових класів пристроїв, що розширюють PC.

З точки зору користувача привабливі такі риси USB:

- простота кабельної системи підключень.
- ізоляція подробиць електричних підключень від користувача.
- самоідентифікація периферії, автоматичний зв'язок пристроїв з драйверами і конфігурація.
- можливість динамічного підключення і реконфігурування периферії.

USB забезпечує обмін даними між хост-комп'ютером і безліччю одночасно доступних периферійних пристроїв. Розподіл пропускну здатності шини між підключеними пристроями планується хостом і реалізується ним з допомогою посилки маркерів. Шина дозволяє підключати, конфігурувати, використовувати і відключати пристрої під час роботи хоста і самих пристроїв - динамічне ("гаряче") підключення і відключення.

Пристрої (Device) USB можуть бути хабами, "функціями" або їх комбінацією. Хаб (Hub) забезпечує додаткові точки підключення пристроїв до шини. "Функції" (Function) USB надають системі додаткові можливості - наприклад підключення до ISDN, цифровий джойстик, акустичні колонки з цифровим інтерфейсом і т.д. Пристрій USB повинен мати інтерфейс USB, що забезпечує підтримку протоколу USB, виконання стандартних операцій

(конфігурація і скидання) і стандартне представлення інформації, що описує пристрій.

Багато пристроїв, що підключаються до USB, мають у своєму складі і "функції" та хаби. Роботою всієї системи USB керує хост-контролер, який є програмно-апаратною підсистемою хост-комп'ютера.

Фізичне з'єднання пристроїв здійснюється по топології багаторівневої зірки. Центром кожної зірки є хаб, кожен кабельний сегмент з'єднує дві точки - хаб з іншим хабом або хаб з функцією. В системі USB має тільки один хост-контролер, розташований у вершині піраміди пристроїв і хабів USB. Хост-контролер інтегрується з кореневим хабом (roothub), що забезпечує одну або кілька точок підключення - портів. Контролер USB, який входить до складу чіпсетів багатьох сучасних системних плат зазвичай має двох / чотирьохпортовий хаб.

Стандарт для високопродуктивної послідовної шини (High Performance Serial Bus), що отримав офіційно назву IEEE 1394, був прийнятий в 1995 році. Метою було створення шини, яка не поступається сучасним дротовим паралельним шинам, при суттєвому здешевленні та підвищенні зручності підключення (за рахунок переходу на послідовний інтерфейс). Стандарт заснований на шині FireWire, використовуваної Apple Computer в якості дешевої альтернативи SCSI в комп'ютерах Macintosh і PowerMac. Назва FireWire ("вогненний провід") тепер застосовується і до реалізацій IEEE 1394, вона співіснує з коротким позначенням 1394.

Переваги FireWire перед іншими послідовними шинами:

- багатофункціональність: шина забезпечує цифровий зв'язок до 63 пристроїв без застосування додаткової апаратури (хабів). Пристрої - цифрові камери, сканери, принтери, камери для відеоконференцій, дис-кові накопичувачі - можуть обмінюватися даними не тільки з PC, але і між собою. FireWire з ініціативи VESA позиціонується і для "домашніх мереж".
- висока швидкість обміну дозволяють навіть на початковому рівні передавати одночасно два канали відео (30 кадрів в секунду) широкоформатної якості та стереоаудіосигнал з якістю CD.
- низька ціна компонентів і кабелю.
- легкість установки і використання. Пристрої автоматично розпізнаються і конфігуруються при включенні/відключенні.

Контрольні запитання:

1. Дайте визначення *форма-фактора* материнської плати.
2. Назвіть сучасні форм-фактори. Які вони мають особливості?
3. Які компоненти комп'ютера розміщуються на материнській платі?
4. Що таке чіп-сет?
5. Дайте визначення *інтерфейсу*.
6. Які інтерфейси існують?
7. У чом особливість інтерфейсу USB?
8. У чом особливість інтерфейсу IEEE-1394?

Тема 2.1 Функції і склад мікропроцесорів. Архітектури мікропроцесорів.

Лекція 1 (2 год.) Функції і склад мікропроцесорів. Режими роботи мікропроцесорів.

План лекції:

1. Функції мікропроцесора.
2. Класифікація мікропроцесорів.
3. Склад мікропроцесорів.
4. Режими роботи мікропроцесорів.

"Мозком" персонального комп'ютера є мікропроцесор, або центральний процесор – CPU (Central Processing Unit). Мікропроцесор виконує обчислення і обробку даних (за винятком деяких математичних операцій, здійснюваних в комп'ютерах, що мають співпроцесор) і, як правило, є найдорожчою мікросхемою комп'ютера. У всіх PC-сумісних комп'ютерах використовуються процесори, сумісні з сімейством мікросхем Intel, але випускаються і проектуються вони як самою Intel, так і компаніями AMD, Cyrix, IDT і Rise Technologies.

Мікропроцесор – це центральний блок персонального комп'ютера, призначений для управління роботою всіх інших блоків і виконання арифметичних і логічних операцій над інформацією.

Мікропроцесор виконує такі основні функції:

- читання і дешифрування команд з основної пам'яті;
- читання даних з основної пам'яті і регістрів адаптерів зовнішніх пристроїв;
- прийом та обробку запитів і команд від адаптерів на обслуговування зовнішніх пристроїв;
- обробку даних і їх запис в основну пам'ять і регістри адаптерів зовнішніх пристроїв;
- вироблення керуючих сигналів для всіх інших вузлів і блоків комп'ютеру.

Всі мікропроцесори поділяють на окремі класи відповідно до їх архітектури, структури і функціонального призначення. На рис. 1 наведено схему класифікації МП за функціональним призначенням.

Мікропроцесори загального призначення призначені для вирішення широкого кола завдань обробки різноманітної інформації. Їх основною областю використання є персональні комп'ютери, робочі станції, сервери та інші цифрові системи масового застосування.

Спеціалізовані мікропроцесори орієнтовані на вирішення специфічних завдань управління різними об'єктами. Містять додаткові мікросхеми (інтерфейси), що забезпечують спеціалізоване використання. Мають особливу конструкцію, підвищену надійність.

Мікроконтролери є спеціалізованими мікропроцесорами, які орієнтовані на реалізацію пристроїв керування, вбудованих у різноманітну

апаратуру. Характерною особливістю структури мікроконтролерів є розміщення на одному кристалі з центральним процесором внутрішньої пам'яті і великого набору периферійних пристроїв.



Рисунок 2.1 – Класифікація мікропроцесорів по функціональному призначенню

Цифрові процесори сигналів (ЦПС) представляють клас спеціалізованих мікропроцесорів, орієнтованих на цифрову обробку вхідних аналогових сигналів. Специфічною особливістю алгоритмів обробки аналогових сигналів є необхідність послідовного виконання ряду команд множення-підсумовування з накопиченням проміжного результату в регістрі-акумуляторі. Тому архітектура ЦПС орієнтована на реалізацію швидкого виконання операцій такого роду. Набір команд цих процесорів містить спеціальні команди MAC (Multiplication with Accumulation), які реалізують ці операції.

Основні компоненти сучасного мікропроцесора:

Ядро – основний компонент процесора, що здійснює виконання команд.

Модуль прогнозування переходу (Branch Predictor). Модуль визначає зміну послідовності виконання команд після переходу, щоб переслати ці команди заздалегідь в декодер команд.

Співпроцесор. Модуль для виконання операцій з нецілими числами (числами з точкою, що плаває).

Кеш-пам'ять. Час доступу до даних модулів набагато швидший, ніж до зовнішньої пам'яті.

Інтерфейсний модуль системної шини. По системній шині CPU надходять команди і дані, які в даному модулі поділяються на два потоки. У разі коли дані та команди виходять від CPU, потоки об'єднуються.

Режими роботи процесора.

Реальний режим. Реальний режим (Real Mode) відповідає можливостям CPU 8086/8088, дозволяючи адресувати трохи більше 1 Мбайт пам'яті. Щоб

підтримати сумісність з раніше розробленими програмами, процесори 286 і навіть Pentium працюють під керуванням операційної системи MS-DOS у реальному режимі і використовують при цьому, звичайно, мінімальні можливості процесора.

Захищений режим. Захищений режим (Protected Mode) з'явився вперше у CPU 80286. У цьому режимі CPU може адресувати до 16 Мбайт фізичної та до 1 Гбайт віртуальної пам'яті. Якщо фізична пам'ять повністю завантажена, то дані, що не помістилися на згадку, розташовуються на вінчестері. Таким чином, CPU працює не з реальними, а з віртуальними адресами, що управляються за допомогою спеціальних таблиць, щоб інформацію можна було знайти (або знову записати). Цю пам'ять називають ще віртуальною пам'яттю, бо фактично вона не існує.

Крім того, у захищеному режимі можлива підтримка мультизадачного режиму (Multitasking). При цьому CPU може виконувати різні програми у виділені кванти часу, відведені кожній із програм (користувачеві ж здається, що програми виконуються одночасно).

Віртуальний режим. Вперше, починаючи з процесора 386, CPU можуть емулювати роботу кількох процесорів 8086 (максимум 256) і, тим самим, забезпечити розрахований на багато користувачів режим так, щоб на одному РС можна було запустити одночасно навіть різні операційні системи. Природно, збільшується і можлива кількість додатків, що виконуються.

Лекція 2 (2 год.) Архітектури мікропроцесорів. Параметри мікропроцесорів.

План лекції:

1. Типи архітектури мікропроцесорів.
2. Параметри мікропроцесорів.
3. Сокет мікропроцесора.

Мікроархітектура мікропроцесора – це апаратна організація і логічна структура мікропроцесора, регістри, керуючі схеми, арифметико-логічні пристрої, запам'ятовуючі пристрої і пристрої, які зв'язують їхні інформаційні магістралі.

Макроархітектура мікропроцесора – це система команд, типи оброблюваних даних, режими адресації і принципи роботи мікропроцесора.

При описі архітектури та функціонування процесора звичайно використовується його подання у вигляді сукупності програмно - доступних регістрів, що утворюють реєстрову або програмну модель. У цих регістрах містяться оброблювані дані (операнди) і керуюча інформація. Відповідно, в реєстрову модель входить група регістрів загального призначення, службовців для зберігання операндів, і група службових регістрів, що забезпечують управління виконанням програми і режимом роботи процесора, організацію звернення до пам'яті (захист пам'яті, сегментна і сторінкова організація та ін.).

Регістри загального призначення утворюють РЗП – **внутрішню реєстрову пам'ять** процесора. Склад і кількість службових реєстрів визначається архітектурою мікропроцесора. Зазвичай в їх склад входять:

Програмний лічильник PC (або CS + IP в архітектурі мікропроцесорів Intel);

- Регістр стану SR (або EFLAGS);
- Регістри управління режимом роботи процесора CR (Control Register);
- Регістри, що реалізують сегментну і сторінкову організацію пам'яті;
- Регістри, що забезпечують налагодження програм і тестування процесора.

Состав пристроїв і блоків, що входять в структуру мікропроцесора, і реалізуються механізми їх взаємодії визначаються функціональним призначенням і областю застосування мікропроцесора.

Архітектура та структура мікропроцесора тісно взаємопов'язані. Реалізація тих чи інших архітектурних особливостей вимагає введення в структуру мікропроцесора необхідних апаратних засобів (пристроїв і блоків) і забезпечення відповідних механізмів їх спільного функціонування. У сучасних мікропроцесорах реалізуються наступні варіанти архітектури.

Всі мікропроцесори можна розділити на групи:

- мікропроцесори типу CISC з повним набором системи команд;
- мікропроцесори типу RISC з усіченим набором системи команд;
- мікропроцесори типу VLIW з надвеликим командним словом;
- мікропроцесори типу MISC з мінімальним набором системи команд і вельми високою швидкодією та ін.

CISC (Complex Instruction Set Computer) – архітектура реалізована в багатьох типах мікропроцесорів, що виконують великий набір різноформатних команд з використанням численних способів адресації. Вони виконують більше 200 команд різного ступеня складності, які мають розмір від 1 до 15 байт і забезпечують більше 10 різних способів адресації. Таке велике різноманіття виконуваних команд і способів адресації дозволяє програмісту реалізувати найбільш ефективні алгоритми вирішення різних завдань. Проте при цьому суттєво ускладнюється структура мікропроцесора, особливо його пристрій управління, що приводить до збільшення розмірів і вартості кристалу, зменшенню продуктивності. У той же час багато команд і способів адресації використовуються досить рідко. Також аналіз кодів програм, які генеруються компіляторами мов вищого рівня, показав, що компілятори з усієї системи команд МП використовують тільки обмежений набір простих команд. Це команди типу "регістр-регістр", "регістр-пам'ять".

RISC (Reduced Instruction Set Computer) – архітектура відрізняється використанням обмеженого набору команд фіксованого формату. Сучасні RISC – процесори зазвичай реалізують близько 100 команд, що мають фіксований формат довжиною 4 байта. Також значно скорочується число використовуваних способів адресації. Зазвичай в RISC - процесорах всі команди обробки даних виконуються тільки з реєстрової або безпосередньою адресацією. Для зменшення кількості звертань до пам'яті RISC-процесори

мають збільшений об'єм внутрішніх регістрів - від 32 до декількох сотень, тоді як в CISC-процесорах кількість регістрів загального призначення переважно становить 8-16.

VLIW (Very Large Instruction Word) – архітектура з'явилася відносно недавно - в 1990 -х роках. Її особливістю є використання дуже довгих команд (до 128 біт і більше), окремі поля яких містять коди, що забезпечують виконання різних операцій. Таким чином, одна команда викликає виконання відразу декількох операцій паралельно в різних операційних пристроях, що входять в структуру мікропроцесора.

Робота процесора полягає і в послідовному виконанні команд з оперативної пам'яті, і в швидкості виконання команд. Чим швидше процесор виконує команди, тим вища продуктивність комп'ютера в цілому. Швидкість роботи процесора залежить від декількох параметрів:

- ступінь інтеграції;
- внутрішня та зовнішня розрядність оброблюваних даних;
- тактова частота;
- кількість інструкцій, які виконуються за один такт;
- пам'ять, до якої може бути адресована CPU;
- обсяг вбудованої кеш-пам'яті;
- кількість рівнів кеш-пам'яті.

Крім того, CPU розрізняються за технологією виробництва, напругою живлення, форм-фактору та ін.

Сокет (socket) процесора – роз'єм, місце на материнській платі комп'ютера, куди вставляється процесор. Процесор, перш ніж він буде встановлений у материнську плату, повинен підходити по сокету.

Все різноманіття сокетів можна розділити на великі групи:

1. Сокети процесорів компанії Intel.
2. Сокети процесорів компанії AMD.

Відрізняються сокети один від одного фізично:

- Кількість контактів
- Типом цих контактів
- Відстанню кріплень для процесорних кулерів
- Власне розміром самого сокету

Кількість контактів – їх може бути 400, 500, 1000 і навіть більше. Як дізнатися? У маркуванні сокету вже міститься вся інформація. Наприклад, процесор Intel Pentium 4 має сокет LGA 775. Так ось 775 - це якраз кількість контактів, а LGA - означає те, що процесор не має контактних ніжок (штирків), вони знаходяться в сокеті материнської плати.

У технологічному плані сокети відрізняються один від одного:

- Наявністю різних додаткових контролерів
- Наявністю чи відсутністю підтримки інтегрованої у процесор графіки (графічне ядро мікропроцесора)

- Більш високими параметрами продуктивності

Процесор чи його сокет впливають на:

- Тип оперативної пам'яті, що підтримується
- Частота шини FSB
- Непрямо (здебільшого - чіпсет) на версію слота PCI-e
- На версію роз'єму SATA (також опосередковано)

Виробники сучасних материнських плат цілеспрямовано залишили за нами можливість змінювати різні пристрої, у тому числі і процесор. Тут те і з'являється таке поняття як сокет, адже з погляду виробників цілком можна було б припаяти процесор прямо до матюки. платі, та й у плані надійності це доцільніше. Але це було, прямо скажемо, спеціально - тобто. для можливого апгрейду системи. Інакше кажучи, захотіли ми замінити процесор на інший - витягли його з сокету і вставили той, який нам треба, звичайно ж з тією поправкою, що він повинен мати такий самий сокет як і у старого процесора. Правду кажучи, саме для можливої модернізації комп'ютерного заліза і існують переважна більшість слотів і роз'ємів, які тільки є на материнській платі.

Контрольні запитання:

1. Дайте визначення мікропроцесора і його основні функції.
2. Наведіть основні компоненти сучасного мікропроцесора.
3. У чому особливість захищеного режиму роботи мікропроцесора?
4. Наведіть типи архітектури мікропроцесора за складом системи команд.
5. Які недоліки та переваги у архітектури CISC і RISC?
6. Що таке *сокет* і що він визначає?

Тема 2.2 Організація і функціонування системи пам'яті.

Лекція 1 (2 год.) Пам'ять і її архітектура. Чинники продуктивності пам'яті.

План лекції:

1. Базові концепції організації пам'яті комп'ютера.
2. Взаємодія пам'яті з процесором.
3. Основні характеристики елементів пам'яті.
4. Види елементів пам'яті.

Максимальний розмір пам'яті, який може використовуватися комп'ютером, визначається системою адресації. Наприклад, 16-розрядний комп'ютер, що генерує 16-розрядні адреси, може мати пам'ять обсягом до $2^{16} = 64$ Кбайт адресованих одиниць зберігання, комп'ютер, команди якого генерують 32-розрядні адреси, може використовувати пам'ять обсягом до $2^{32} = 4$ Гбайт одиниць, що адресуються, а для комп'ютерів з 40-розрядними адресами доступна пам'ять обсягом до $2^{40} = 1$ Тбайт одиниць, що адресуються. Кількість одиниць пам'яті, що адресуються, визначає розмір її адресного простору.

Зазвичай пам'ять розробляється з огляду на те, що дані виймаються і зчитуються не байтами, а словами. Саме поняття **довжини слова** найчастіше визначається як кількість бітів, що зберігаються або зчитуються за одне звернення до пам'яті.

З боку системи пристрій можна розглядати як чорну скриньку. Пересилання даних між пам'яттю та процесором виконується за допомогою двох регістрів процесора, зазвичай званих MAR (Memory Address Register – регістр адреси) та MDR (Memory Data Register – регістр даних). Якщо регістр MAR містить k бітів, а регістр MDR – n бітів, пам'ять може містити до 2^k одиниць зберігання, що адресуються. За один цикл звернення до пам'яті між нею та процесором пересилається n біт даних. Дані передаються шиною процесора, що має k адресних ліній і n ліній даних. Крім того, шина містить лінії для керування передачею даних $R/\overline{W}$ (READ/ $\overline{WRITE}$) та MFC (Memory Function Compelled). Можуть використовуватися й інші лінії, за допомогою яких задається кількість даних, що пересилаються.

Щоб вважати дані з пам'яті, процесор спочатку завантажує адресу в регістр MAR і встановлює лінію $R/\overline{W}$ в 1. У відповідь пам'ять поміщає дані лінії даних і підтверджує цю дію активізацією сигналу MFC. Після отримання сигналу MFC процесор завантажує дані з адресних ліній регістр MDR.

Для того щоб записати дані в пам'ять, процесор завантажує адресу в регістр MAR, а дані в регістр MDR і встановлює лінію $R/\overline{W}$ в 0, вказуючи таким чином, що виконується операція запису.

Якщо в операціях читання здійснюється звернення за послідовними адресами, може бути виконана операція блокового пересилання, при якій пам'яті передається лише одна адреса - адреса першого байта блоку даних.

Швидкодія пам'яті характеризується інтервалом часу між ініціюванням операції та її завершенням, наприклад часом між сигналами читання та MFC. Цей час визначають як **час доступу до пам'яті**. Ще однією важливою характеристикою швидкодії пам'яті є **цикл пам'яті** – мінімальний проміжок часу між моментами початку двох послідовних операцій із пам'яттю, наприклад, між двома послідовними операціями читання. Цикл пам'яті зазвичай трохи більше часу доступу (це залежить від особливостей реалізації пристрою).

У пам'яті, що називається **пам'яттю з довільним доступом** (Random Access Memory, RAM), на звернення за будь-якою адресою йде один і той же час. Цим RAM-пам'ять відрізняється від пристроїв з послідовним або частково-послідовним доступом, таких як магнітні диски або стрічки. Час доступу останніх залежить від адреси або розташування даних.

Для реалізації основної пам'яті комп'ютера використовують напівпровідникові інтегральні схеми.

Основними характеристиками елементів (мікросхем) пам'яті є:

- тип
- ємність
- розрядність
- швидкодія
- часова діаграма.

В ідеалі пам'ять має бути швидкою, великою та дешевою. Для реалізації дуже швидкої пам'яті краще використовувати мікросхеми SRAM (Static RAM). Однак вони досить дорогі, оскільки кожен осередок такої пам'яті містить шість транзисторів, через що на одній мікросхемі неможливо розмістити багато осередків. Таким чином, велику пам'ять на основі мікросхем SRAM недоцільно створювати із суто економічних міркувань. Для неї застосовуються набагато дешевше мікросхеми динамічна RAM (Dynamic RAM, DRAM), синхронна DRAM (SDRAM) і Rambus DRAM, з дуже простими осередками, об'єднані в модулі великого об'єму SIMM, DIMM і RIMM. Щоправда, працює така пам'ять повільніше за SRAM.

Час, необхідний читання (запису) даних, що зберігаються за випадковою адресою, називається **часом доступу** (Access time). Для сучасних мікросхем час доступу становить близько 3-6 нс, а тактова частота шини пам'яті – 800-2000 МГц і вище.

Хоча в комп'ютер за цілком розумну ціну можна встановити сотні мегабайт динамічної пам'яті, цього все одно недостатньо для сучасних програм, що обробляють величезні обсяги даних. Тому для збільшення адресного простору пам'яті використовуються зовнішні пристрої, насамперед **магнітні диски**. Сучасні диски мають дуже велику ємність і зазвичай прийнятну ціну, тому вони досить інтенсивно застосовуються в комп'ютерних системах. Однак функціонують магнітні диски значно повільніше напівпровідникової пам'яті. З усього сказаного випливає такий висновок: дуже великий обсяг пам'яті за прийнятною ціною можна отримати лише як жорсткого

диска. Для реалізації основної пам'яті великого обсягу підходить технологія DRAM/SDRAM. А мікросхеми SRAM можна використовувати для створення дуже швидких пристроїв невеликого об'єму, таких як кеш-пам'ять.

Як правило, у комп'ютері використовуються всі три типи пам'яті. Отже, систему пам'яті комп'ютера можна уявити як ієрархію. Найшвидше здійснюється доступ до даних, що зберігаються в регістрах процесора. Тому, якщо розглядати ці регістри як складову ієрархії пам'яті, то з точки зору швидкодії вони розташовуються на самому верху, хоча за обсягом становлять мізерну частину всієї пам'яті комп'ютера.

На наступному рівні ієрархії знаходиться порівняно невеликий обсяг пам'яті, який можна реалізувати прямо в мікросхемі процесора. Ця пам'ять, звана кешем процесора, містить копії команд і даних, що зберігаються у зовнішній по відношенню до процесора та набагато більшої основної пам'яті. Зазвичай комп'ютер має два рівні кеш-пам'яті.

Нижче ієрархії розташовується основна пам'ять. Вона досить велика і реалізується з урахуванням мікросхем динамічної пам'яті, зазвичай, як модулів DIMM і RIMM. Основна пам'ять значно більша і набагато повільніша за кеш. У типовому комп'ютері час доступу до основної пам'яті вдесятеро більший за час доступу до кешу L1.

Дискові пристрої надають, можна сказати, величезний обсяг недорогої пам'яті, але порівняно з напівпровідниковими пристроями вони дуже повільні.

Для виконання програм виняткове значення має швидкість доступу до пам'яті. Ідея управління ієрархічною системою пам'яті полягає в тому, щоб перемістити команди та дані, які будуть використовуватися найближчим часом, якомога ближче до процесора.

Лекція 2 (2 год.) Організація кеш-пам'яті. Віртуальна пам'ять. Апаратні засоби підтримки віртуальної пам'яті.

План лекції:

1. Функції кеш-пам'яті.
2. Рівні кеша.
3. Віртуальна адреса.
3. Організація віртуальної пам'яті.

Зазвичай процесор обробляє команди та дані швидше, ніж вони вибираються з пам'яті. Тому цикл доступу до пам'яті є вузьким місцем системи. Одним із способів скорочення часу доступу до пам'яті може стати використання **кеш-пам'яті**. Це швидка пам'ять невеликого обсягу, розташована між порівняно повільною основною пам'яттю комп'ютера та процесором. У ній зберігаються поточні фрагменти програми та її даних.

Первинний кеш розміщується на мікросхемі процесора і називається кешем першого рівня (L1). Вторинний кеш має більший обсяг, розташовується

між первинним кешем та рештою пам'яттю і називається кешем другого рівня (L2). Для його реалізації зазвичай використовуються мікросхеми SRAM.

Типовий комп'ютер містить кеш першого рівня, що розташовується на мікросхемі процесора, і зовнішній по відношенню до процесора кеш другого рівня, дещо більшого обсягу. Однак, це не завжди так. Буває, що мікросхема процесора взагалі не містить кешу або, навпаки, містить кеш обох рівнів.

Ще однією важливою концепцією, пов'язаною з організацією пам'яті, є концепція **віртуальної пам'яті**. Досі ми припускали, що адреси, що генеруються процесором, повністю відповідають фізичним осередкам пам'яті. Проте практично це завжди так. З певних причин дані можуть зберігатися не за адресами, які задані в програмі. Схеми керування пам'яттю перетворюють вказану в програмі адресу на іншу адресу, що використовується для доступу до фізичної пам'яті. У такому випадку згенеровану процесором адресу називають віртуальною або логічною. Віртуальний адресний простір певним чином відображається на фізичну пам'ять, де зберігаються дані. Відображення виконується за допомогою спеціальних схем керування пам'яті, які часто називають *блоком управління пам'яттю*.

Віртуальна пам'ять призначена збільшення видимого комп'ютером обсягу фізичної пам'яті. Віртуальний адресний простір і обсяг розміщених у ньому даних можуть бути настільки великими, наскільки дозволяють можливості адресації процесора, що використовується. Причому кожен конкретний момент лише частина цього адресного простору відображається на фізичну пам'ять. Інші віртуальні адреси відповідають пристроям масової пам'яті – зазвичай магнітним дискам. Коли програма потребує доступу до даних, адреси яких не відображаються на реальну пам'ять, блок керування пам'яттю змінює функцію відображення і пересилає дані з диска в основну пам'ять. Тому в кожному циклі звернення до пам'яті система обробки адрес визначає, чи знаходиться адресована інформація в основній пам'яті комп'ютера. Якщо так, відбувається звернення до відповідного слова пам'яті та виконання програми продовжується. Якщо ні, з диска на згадку пересилається сторінка, що містить потрібне слово. Ця сторінка замінює в пам'яті якусь іншу сторінку, яка поки що не потрібна програмі. Оскільки на пересилання сторінок між пам'яттю та диском витрачається деякий час, при частому переміщенні сторінок швидкість роботи комп'ютера знижується. Але якщо вибір замінюваних сторінок виконується продумано, ймовірність того, що необхідні сторінки опиняться в основній пам'яті, зросте.

Контрольні запитання:

1. Як процесор взаємодіє з пам'яттю?
2. Що таке *час доступу* до пам'яті? Що таке *цикл пам'яті*?
3. Основні характеристики елементів пам'яті.
4. Які види пам'яті використовують у комп'ютері?
5. Які функції кеша?
6. Як віртуальна пам'ять організує адресний простір комп'ютера?

Тема 2.3 Накопичувачі на жорсткому магнітному диску. Зовнішні запам'ятовуючі пристрої.

Лекція 1 (2 год.) Фізична будова жорсткого диска. Продуктивність жорсткого диска. Інтерфейси підключення жорсткого диска.

План лекції:

1. Логічна структура жорсткого диску.
2. Фізична будова жорсткого диску.
3. Основні характеристики жорстких дисків.

У накопичувачах на жорстких дисках дані записуються і зчитуються універсальними головками читання / запису з концентричних кіл обертових магнітних дисків (*доріжок*), розбитих на *сектори* ємністю 512 байт (рис. 1).

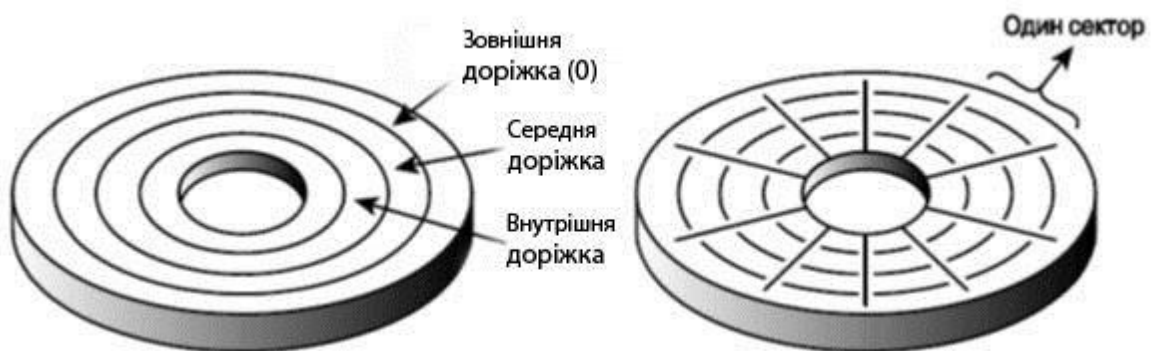


Рисунок 2.2 – Доріжки і сектори накопичувача на жорстких дисках.

У накопичувачах зазвичай встановлюється декілька дискових пластин і дані записуються на обох сторонах кожної з них. У більшості накопичувачів є щонайменше два або три диска (що дозволяє виконувати запис на чотирьох чи шести сторонах), але існують також пристрої, що містять до 11 і більше дисків. Однотипні (однаково розташовані) доріжки на всіх сторонах дисків об'єднуються в *циліндр* (рис. 2.2). Для кожної сторони диска передбачена своя доріжка читання / запису, але при цьому все головки змонтовані на загальному стрижні, або *приводі*. Тому головки не можуть переміщатися незалежно один від одного, тобто рухаються тільки синхронно.

Незважаючи на те що швидкість обертання може змінюватися, сучасні пристрої розкручують пластини до 4200, 5400 7200, 10000 і навіть 15000 об/хв. Деякі диски малих формфакторів з метою економії електроенергії розкручуються за все до 4200 об / хв, в той час як високошвидкісні можна зустріти в основному в робочих станціях і серверах, де підвищена ціна, а також додатковий шум і тепловиділення не грають вирішальної ролі. Висока швидкість обертання в поєднанні зі швидкісним механізмом подачі головок і великою кількістю секторів на доріжці – ось головні чинники, що визначають загальну продуктивність жорсткого диска.

При нормальній роботі жорсткого диска головки читання / запису не торкаються (і не повинні торкатися!) дисків. Але при виключенні живлення і зупинці дисків вони опускаються на поверхню. Під час роботи пристрою між головкою і поверхнею диска, що обертається утворюється дуже малий

повітряний зазор (повітряна подушка). Якщо в цей зазор потрапить порошинка або відбудеться струс, головка "зіткнеться" з диском, що обертається "на повному ході". Якщо удар буде досить сильним, відбудеться поломка головки. Наслідки цього можуть бути різними – від втрати кількох байтів даних до виходу з ладу всього накопичувача. Тому в більшості накопичувачів поверхні магнітних дисків легірують і покривають спеціальними мастилами, що дозволяє пристроям витримувати щоденні "злети" і "приземлення" головок, а також більш серйозні потрясіння.

У деяких сучасних накопичувачах замість конструкції CSS (Contact Start Stop) використовується механізм завантаження / розвантаження, який не дозволяє голівкам входити в контакт з жорсткими дисками навіть при відключенні живлення накопичувача. У механізмі завантаження / розвантаження використовується похила панель, розташована безпосередньо над зовнішньою поверхнею жорсткого диска. Коли накопичувач вимкнений або знаходиться в режимі економії споживаної потужності, головки з'їжджають на цю панель. При подачі електроенергії головки розблокуються тільки тоді, коли швидкість обертання жорстких дисків досягне потрібної величини. Потік повітря, створюваний при обертанні дисків (аеростатичний підшипник), дозволяє уникнути можливого контакту між головою і поверхнею жорсткого диска.

Існує безліч типів накопичувачів на жорстких дисках, але практично всі вони складаються з одних і тих же основних вузлів. Конструкції цих вузлів, а також якість використовуваних матеріалів можуть відрізнятися, але їх основні робочі характеристики і принципи функціонування однакові. Основні елементи конструкції типового накопичувача на жорсткому диску (рис. 9.6) перераховані нижче:

- диски;
- голівки читання / запису;
- механізм приводу головок;
- двигун приводу дисків;
- друкована плата зі схемами управління;
- кабелі і роз'єми;
- елементи конфігурації (перемички і перемикачі).

Диски, двигун приводу дисків, головки і механізм приводу головок зазвичай розміщуються в герметичному корпусі, який називається HDA (Head Disk Assembly – блок головок і дисків). Зазвичай цей блок розглядається як єдиний вузол; його майже ніколи не розкривають. Інші вузли, що не входять в блок HDA (друкована плата, лицьова панель, елементи конфігурації і монтажні деталі), є знімними.

Диски. Накопичувач на жорстких магнітних дисках містить кілька дисків (пластин). Протягом багатьох років жорсткі диски для ПК випускалися в декількох формфакторах. Як правило, фізичні розміри жорстких дисків виражаються в розмірі використовуваних пластин.

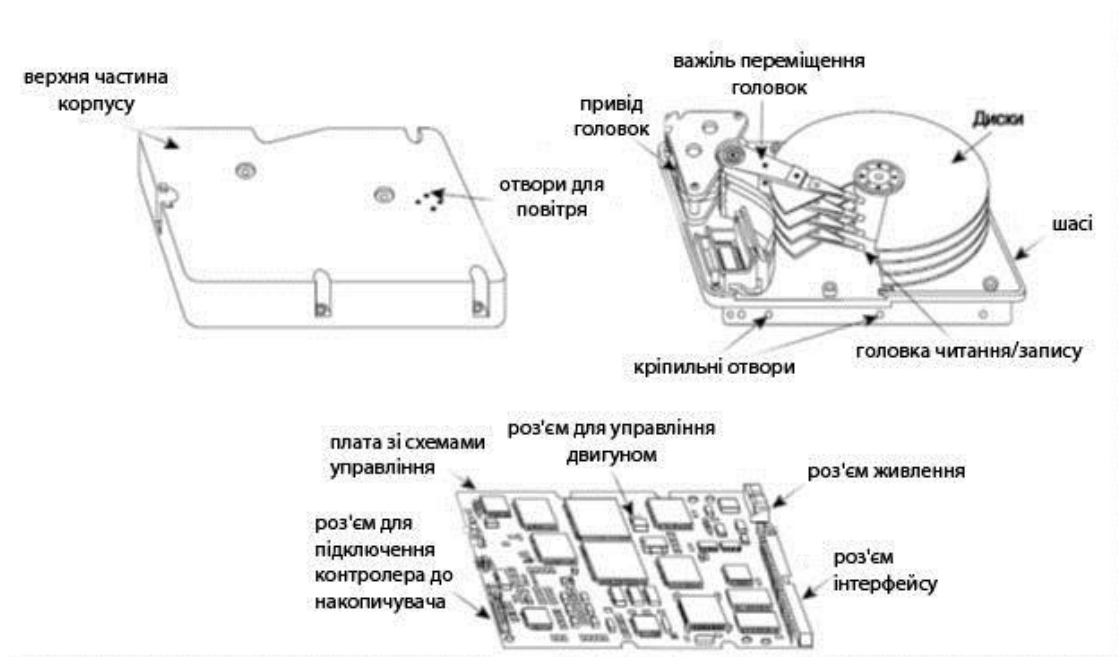


Рисунок 2.3 – Основні вузли накопичувача на жорсткому диску.

У більшості накопичувачів встановлюється мінімум два диска, хоча в деяких малих моделях буває і по одному. Кількість дисків обмежується фізичними розмірами накопичувача, а саме – висотою його корпусу. Найбільша кількість дисків в накопичувачах формату 3,5 дюйма – 12.

Головки читання / запису. У накопичувачах на жорстких дисках для кожної зі сторін кожного диска передбачена власна головка читання / запису. Всі головки змонтовані на загальному рухомому каркасі і переміщуються одночасно.

Конструкція каркаса з головками досить проста. Кожна головка встановлена на кінці важеля, закріпленого на пружині і злегка притискає її до диска. Мало хто знає про те, що диск як би затиснутий між парою головок (зверху і знизу). І якби це не спричинило за собою ніяких наслідків, можна було б провести невеличкий експеримент: відкрити накопичувач і підняти пальцем верхню головку. Як тільки б ви її відпустили, вона повернулася б в початкове положення (те ж саме відбулося б і з нижньою головкою).

Коли накопичувач вимкнений, головки торкаються дисків під дією пружин. При розкручуванні дисків аеродинамічний тиск під головками підвищується, і вони відриваються від робочих поверхонь ("злітають"). Коли диск обертається на повній швидкості, зазор між ним і головками може становити 0,5-5 мікродюймів і навіть більше.

Механізми приводу головок. Мабуть, ще більш важливою деталлю накопичувача, ніж самі головки, є механізм, який встановлює їх в потрібне положення; він називається приводом головок. Саме з його допомогою головки переміщуються від центру до країв диска і встановлюються на заданий циліндр. Існує багато конструкцій механізмів приводу головок, але їх можна розділити на два основних типи: з кроковим двигуном; з рухомою котушкою.

Тип приводу багато в чому визначає швидкодію і надійність накопичувача, достовірність зчитування даних, його температурну стабільність, чутливість до вибору робочого положення і вібрацій. Скажемо відразу, що накопичувачі з приводами на основі крокових двигунів набагато менш надійні, ніж пристрої з приводами від рухомих котушок.

До основних характеристик накопичувачів на жорстких дисках належать: ємність; швидкодія; надійність; вартість.

Швидкодію накопичувача можна оцінити за двома параметрами: швидкість передачі даних; середньостатистичний час пошуку.

Середній час доступу. Середнім часом доступу до даних називається сума середнього часу позиціонування і часу очікування. Середній час доступу зазвичай виражається в мілісекундах.

Цей показник характеризує час, необхідний накопичувачу для звернення до довільно розташованого сектору.

Середній час позиціонування. Це час зазвичай вимірюється в мілісекундах (мс); воно необхідне для переміщення головки від одного циліндра до іншого на будь яку довільну відстань. Один із способів, що дозволяє визначити цю величину, складається в багаторазовому виконанні операцій пошуку випадкової доріжки і наступному розподілі витраченого часу на кількість виконаних операцій. Цей метод дозволяє обчислити середній час, необхідний для виконання однієї операції пошуку доріжки.

Час очікування. Часом очікування називається середній час (в мілісекундах), необхідний для переміщення головки до зазначеного сектору після досягнення певної доріжки. В середньому ця величина дорівнює половині часу, який потрібен для одного обороту жорсткого диска.

При збільшенні частоти обертання диска вдвічі час очікування зменшиться наполовину.

Жорсткий диск підключається до системної плати за допомогою IDE-кабелю (у нових моделях ПК використовується кабель SATA – Serial Advanced Technology Attachment). Кабель підключається до одного з двох прямокутних роз'ємів IDE, встановлених в передній частині материнської плати біля лицьової сторони системного блоку. IDE-інтерфейс дозволяє підключити до чотирьох жорстких дисків – по два на кожен IDE- канал. На кожному IDE-кабелі є три роз'єми : два розташовані близько один до одного (до них підключаються жорсткі диски), останній роз'єм підключається в гніздо одного з двох IDE -контролерів на материнській платі.

Лекція 2 (2 год.) SDD, CD, DVD-ROM. Пристрій, інтерфейс, принцип роботи.

План лекції:

1. Твердотільні накопичувачі.
2. Принцип збереження інформації на CD-ROM.
3. CD-R, CD-RW.
4. DVD-ROM.

Пристрої на базі енергонезалежної флеш-пам'яті називають твердотільними накопичувачами або **SSD-дисками** (Solid State Disk)

Основними елементами SSD є:

PCB – друкована плата.

NAND-flash – флеш-пам'ять NAND; відповідає за зберігання даних.

NAND-controller – контролер пам'яті; виступає в ролі посередника між носієм та системою, і є процесором, що відповідає за продуктивність SSD.

DRAM – кеш (присутня не у всіх моделях SSD); виступає тимчасовим сховищем невеликого обсягу даних та дозволяє стабілізувати знос пам'яті, а також прискорити доступ до файлів.

HOST Interface – інтерфейс підключення; тип з'єднання та протокол, через які SSD з'єднується з вашою системою.

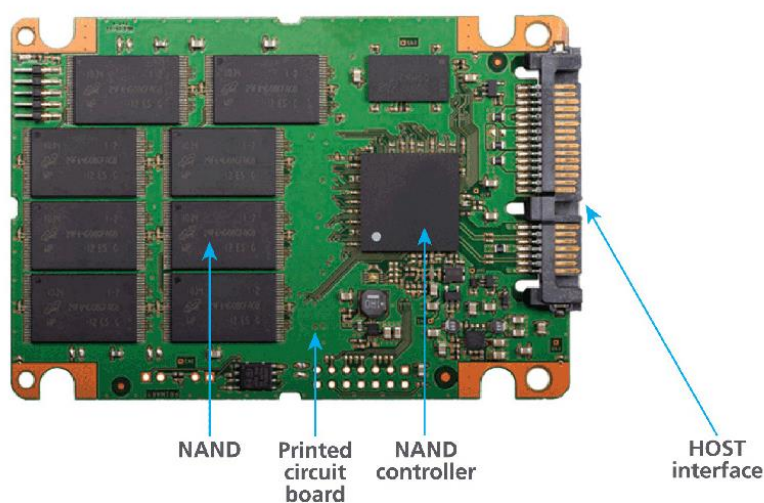


Рисунок 2.4 – Твердотільний накопичувач.

Переваги SSD-дисків:

1. висока швидкодія;
2. безшумність (немає деталей, що рухаються, і ці диски не гріються при своїй роботі, тому охолоджуючі кулери рідше включаються і працюють не так інтенсивно (створюючи шум));
3. ударо- та віброміцність
4. мала вага
5. низьке енергоспоживання.

Недоліки SSD-дисків:

1. висока вартість
2. обмежена кількість циклів перезапису (звичайний, середній SSD-диск на основі флеш-пам'яті з технологією MLC здатний виробити приблизно 10 000 циклів читання запису інформації. Більш дорогий тип пам'яті SLC вже може в 10 разів довше прожити (100 000 циклів перезапису));
3. неможливість відновлення віддаленої інформації.

Усі **компакт-диски** мають бути 120 мм у діаметрі та 1,2 мм у товщину, а діаметр отвору в середині має становити 15 мм. Аудіо компакт-диски були першими засобами зберігання цифрової інформації, що вийшли на масовий ринок. Передбачається, що вони використовуватимуться протягом ста років.

Компакт-диск виготовляється з використанням потужного інфрачервоного лазера, який випалює отвори діаметром 0,8 мікрона в спеціальному скляному майстер-диску. З цього майстер-диску робиться шаблон із виступами у тих місцях, де лазер пропалив отвори. У шаблон вводиться рідка смола (полікарбонат), і таким чином виходить компакт-диск з тим же набором отворів, що й у скляному диску. На смолу наноситься дуже тонкий шар алюмінію, який у свою чергу покривається захисним лаком. Після цього наклеюється етикетка. Поглиблення у нижньому шарі смоли називаються **лунками** (pits), а рівні простори між лунками – **площадками** (lands).

Під час відтворення лазерний діод невеликої потужності світить інфрачервоним світлом з довжиною хвилі 0,78 мікрона на лунки і майданчики, що змінюють один одного. Лазер знаходиться на тій стороні диска, на яку нанесений шар смоли, тому лунки для лазера перетворюються на виступи на рівній поверхні. Так як лунки мають висоту в чверть довжини світлової хвилі лазера, довжина світлової хвилі, відбитої від виступу, становить половину довжини світлової хвилі, відбитої від навколишнього виступу рівної поверхні.

В результаті, якщо світло відбивається від виступу, фотодетектор програвача отримує менше світла, ніж при відображенні від майданчика. Саме в такий спосіб програвач відрізняє лунку від майданчика.

Лунки та майданчики записуються по спіралі. Запис починається на деякій відстані від отвору в центрі диска та просувається до краю, займаючи 32 мм диска.

Пристрій для читання компакт-дисків зі швидкістю, що позначається як 32x (4 915 200 байт/с), незрівнянно по швидкодії з магнітним диском SCSI-2 (10 Мбайт/с), хоча багато пристроїв для читання компакт-дисків використовують інтерфейс SCSI (крім того, застосовується інтерфейс IDE).

Звідси зрозуміло, що компакт-диски за продуктивністю значно поступаються магнітним дискам, незважаючи на їхню велику ємність.

Розвиток технологій, а також попиту, і став причиною народження **DVD-дисків**. Спочатку брєвіатура DVD розшифровувалась як Digital Video Disk

(Цифровий відеодиск), зараз вона офіційно перетворилася на Digital Versatile Disk (цифровий багатоцільовий диск). DVD-диски загалом схожі на компакт-диски. Як і звичайні компакт-диски, вони мають 120 мм у діаметрі, створюються на основі полікарбонату та містять лунки та майданчики, які освітлюються лазерним діодом та зчитуються фотодетектором. Проте є кілька відмінностей:

менший розмір лунок (0,4 мікрона замість 0,8 мікрона, як у звичайного компакт-диска);

більш щільна спіраль (0,74 мікрона між доріжками замість 1,6 мікрона);

червоний лазер (з довжиною хвилі 0,65 мікрона замість 0,78 мікрона).

У сукупності ці удосконалення дали семиразове збільшення ємності (до 4,7 Гбайт). Зчитувальний пристрій для DVD 1x працює зі швидкістю 1,4 Мбайт/с (швидкість роботи зчитувального пристрою для компакт-дисків становить 150 Кбайт/с). На жаль, через перехід до червоного лазера DVD-програвачам був потрібен другий лазер для читання існуючих музичних та комп'ютерних компакт-дисків, що трохи збільшувало їхню складність і вартість.

Контрольні запитання:

6. Що називають **доріжкою** вінчестера?
7. Що називають **сектором** вінчестера??
8. Що називають **циліндром** вінчестера?
9. Що таке час **очікування** сектора?
10. Що таке SSD-диск? Що собою являє елементарна комірка цього накопичувача?
11. Назвіть основні переваги та недоліки SSD-дисків.
12. Яким чином в CD-R формуються лунки й площадки?
13. Як в CD-ROM записуються 1, а як 0?

Тема 2.4 Відеоадаптери і монітори. Аудіосистема комп'ютера.

Лекція 1 (1 год.) Відеоадаптери. Можливості відеосистеми при використанні PCI-Express.

План лекції:

1. Функції та склад відеоадаптера.
2. Чинники, що впливають на продуктивність відеоадаптера.
3. Особливості інтерфейса PCI-Express.

Відеоадаптер (відеокарта, відеоплата) – це пристрій, що здійснює інтерфейс з комп'ютером при підключенні монітора. Кожен комп'ютер має відеокарту, виключаючи ті, в яких вся необхідна електроніка вбудована прямо в материнську плату (в цьому випадку можна встановити нову відеокарту, але стару доведеться попередньо відключити).

Фізично відеокарта являє собою багатошарову друкарську плату, на якій змонтовані мікросхеми, конденсатори і деякі інші деталі, а також роз'єми для підключення монітора (зазвичай кількох), і, у багатьох випадках, телевізора. Окремі моделі мають відеовхід, виконаний у вигляді роз'єму RCA, а іноді він поєднується з відеовиходом.

Функціонально відеоадаптер складається з декількох обов'язкових блоків:

- графічний процесор, який називають також графічним чіпсетом (chipset – набір мікросхем, комплект чіпів);
- відеоконтролер – це пристрій, що відповідає за виведення зображення з відеопам'яті, регенерацію її вмісту, формування сигналів розгортки для монітора і обробку запитів центрального процесора;
- мікросхеми відеопам'яті, в які записані BIOS відео, екранні шрифти, службові таблиці і т.п. ПЗП не використовується відеоконтролером прямо – до нього звертається тільки центральний процесор, і в результаті виконання ним програм з ПЗП відбуваються звернення до відеоконтролеру і відеопам'яті. ПЗП необхідно лише для початкового запуску адаптера і роботи в режимі MS DOS; операційні системи з графічним інтерфейсом (Windows або OS/2) не використовують ПЗП для управління адаптером;
- цифро-аналогові перетворювачі (RAMDAC);
- роз'єми.

Процесор відеокарти. Перші комп'ютерні відеокарти були побудовані за принципом кадрового буфера, згідно з яким власне зображення формувалося центральним процесором комп'ютера і програмним забезпеченням, а карта відповідала лише за зберігання (в буфері пам'яті) і виведення з певною частотою окремих кадрів на монітор. Однак підвищення вимог до якості зображення, пов'язане з появою складних тривимірних комп'ютерних ігор і професійних конструкторських систем, привело до необхідності створення спеціалізованого процесора, який би займався виключно формуванням (точніше, розрахунком) зображення. При цьому

центральный процесор комп'ютера звільнився практично від всіх функцій, пов'язаних з побудовою зображення. Сучасні графічні процесори по складності не поступаються центральному процесорам, і більш того, у багатьох популярних моделях використовуються технології, ще не знайшли застосування в центральних процесорах.

Спочатку дані в цифровому вигляді з шини потрапляють в відеопроцесор, де вони починають оброблятися. Після цього оброблені цифрові дані направляються в відеопам'ять, де створюється образ зображення, яке повинно бути виведено на дисплей.

Для виключення конфліктів при зверненні до пам'яті з боку відеоконтролера і центрального процесора перший має окремий буфер, який у вільний від звернень ЦП час заповнюється даними з відеопам'яті.

Перш ніж стати зображенням на моніторі, двійкові цифрові дані обробляються центральним процесором, потім через шину даних направляються в відеоадаптер, де вони обробляються і перетворюються в аналогові дані і вже після цього направляються в монітор і формують зображення. Потім, все ще в цифровому форматі, дані, що утворюють образ, передаються в RAMDAC, де вони конвертуються в аналоговий вигляд, після чого передаються в монітор, на якому виводиться потрібне зображення.

На продуктивність графічної підсистеми впливають кілька факторів:

- Кількість універсальних шейдерних процесорів (CUDA-ядер);
- Підтримуваних API і технологій;
- Номінальної/динамічної частоти графічного ядра, МГц;
- Частоти пам'яті (ефективної), МГц;
- Обсяг пам'яті, ГБ;
- Типу пам'яті;
- Ширини шини пам'яті, біт;
- Пропускної здатності пам'яті, ГБ/с;
- Типу шини;
- Максимальної роздільної здатності.

Відеопам'ять служить для зберігання зображення – грає роль кадрового буфера. Центральний процесор комп'ютера направляє відеодані в цю спеціалізовану пам'ять, а потім графічний процесор відеокарти зчитує звідти отриману інформацію.

Природно, для забезпечення ефективної передачі даних є важливою пропускну здатність відеопам'яті, характеризуєма розрядністю, ефективна частотою роботи шини, по якій передаються дані з відеопам'яті до графічного процесору і латентністю (latency – час затримки при передачі даних) мікросхем пам'яті.

Обсяг відеопам'яті, встановленої на карті, важливий не стільки для прискорення швидкості роботи самої карти, яка визначається, в значній мірі, пропускну здатністю всієї відеосистеми, скільки для роботи з тривимірними зображеннями з високими дозволами і великою глибиною кольору. Максимально можливий повний дозвіл відеокарти – $A \times B \times C$, де A – кількість

точок по горизонталі, В – по вертикалі, і С – кількість розрядів, необхідна для зберігання можливих кольорів кожної точки. Наприклад, для дозволу 640x480x16 досить 256 Кб, для 800x600x256 – 512 Кб, для 1024x768x65536 (інше позначення – 1024x768x64k) – 2 Мб, і т.д. Оскільки для зберігання кольорів відводиться ціла кількість розрядів, кількість кольорів завжди є ступенем двійки (16 кольорів – 4 розряду, 256 – 8 розрядів, 64к – 16, і т.д.).

Інтерфейс PCI-Express (PCI-E) – це засіб взаємодії, що складається з контролера шини та відповідного слоту (рис. 1) на материнській платі.



Рисунок 2.5 – Роз’єми PCI-Express 3.0 на материнській платі.

Даний високопродуктивний протокол використовується для підключення відеокарти в систему. Відповідно, на материнській платі присутній відповідний слот PCI-Express, куди і встановлюється відеоадаптер. Цей протокол прийшов на зміну інтерфейсу AGP, коли даного інтерфейсу, просто кажучи «перестало вистачати».

Незважаючи на те, що назви і PCI і PCI-Express дуже схожі, принципи взаємодії у них кардинально відрізняються. У PCI-Express використовується лінія – двонаправленого послідовного з’єднання, типу "точка-точка", даних ліній може бути декілька. У випадку з відеокартами і материнськими платами (не враховуємо Cross Fire і SLI), які підтримують PCI-Express x16 (тобто більшість), можна запросто здогадатися, що таких ліній 16, досить часто на материнських платах з PCI E 1.0, можна було спостерігати другий слот x8, для роботи в режимі SLI або Cross Fire.

Для Інтерфейсу **PCI-Express 1.0** пропускна здатність становить 2,5 Гбіт / с. Ці дані потрібні нам, щоб відстежувати зміни цього параметра в різних версіях PCI-E.

Далі, версія 1.0 еволюціонувала в PCI-E 2.0. В результаті даного перетворення, ми отримали в два рази більшу пропускну здатність, тобто 5 Гбіт / с, але хотілося б відзначити, що в продуктивності графічні адаптери, особливо не виграли, так як це просто версія інтерфейсу. Велика частина продуктивності залежить від самої відеокарти, версія інтерфейсу може тільки

незначно поліпшувати або гальмувати передачу даних (в даному випадку «гальмування» немає, і присутній непоганий запас).

Точно так само в 2010 році, «з запасом», був розроблений інтерфейс PCI-E 3.0, на даний момент він використовується у всіх нових системах.

У версії PCI-E 3.0, пропускна здатність була збільшена в два рази в порівнянні з версією 2.0. Також там було вироблено чимало технічних змін.

Лекція 2 (2 год.) Основні принципи побудови моніторів. Основні параметри моніторів.

План лекції:

1. Класифікація моніторів.
2. Принцип формування зображення основними типами моніторів.
3. Основні параметри моніторів.

Монітор – це конструктивно закінчений пристрій, призначений для візуального відображення інформації.

Монітор складається з екрану (дисплея), блока живлення, плат керування, корпусу.

Інформація для відображення на моніторі надходить з електронного пристрою, що формує відеосигнал (у комп'ютері відеокарта).

Класифікація моніторів здійснюється:

- за типом екрана;
- за співвідношенням сторін;
- за типом відеоадаптера (формату);
- за типом інтерфейсного кабелю.

Класифікація моніторів по типу екрана:

– ЕПТ - монітор на основі електронно-променевої трубки (англ. Cathode Ray Tube, CRT);

– РК - рідкокристалічний монітор (англ. Liquid Crystal Display, LCD);

– OLED – монітор на основі технології OLED (англ. Organic Light-Emitting Diode – органічний світловипромінюючий діод, ОСІД); - Сенсорний (резистивний, ємнісний) - монітор з можливістю введення даних;

– проектор – відеопроєктор та екран.

Класифікація моніторів за співвідношенням сторін та розмірами. Відношення сторін "горизонталь: вертикаль" (в умовних одиницях):

- 4:3 (стандартний);
- 16:9 чи 16:10 (широкоформатні);
- 25:16 та ін.

Розмір діагоналі екрану в дюймах (1 дюйм = 2,54 см):

15 (38 см); 17 (43 см); 19 (48 см); 21 (53 см); 22 (56 см); 24 (61 см); 26 (66 см); 30 (76 см); 32 (81 см).

Класифікація моніторів типу відеоадаптера (формату):

- VGA (640x480);

- SVGA (800x600);
- HD 720 (1280x720);
- Full HD 1080 (1920x1080) та ін.

ЕПТ монітори

Довгий час найбільш поширеними пристроями відображення інформації були монітори на основі ЕЛТ. Принцип дії таких моніторів мало відрізняється від принципу дії звичайного телевізора і полягає в тому, що пучок електронів, що випускається електронною гарматою, потрапляючи на екран, покритий люмінофором, викликає його свічення. На шляху пучка електронів зазвичай знаходяться додаткові електроди: модулятор, що регулює інтенсивність пучка електронів і пов'язану з нею яскравість зображення, фокусуючий електрод, що визначає розмір світлової плями, а також розміщені на горловині ЕЛТ котушки системи відхилення, що дозволяють змінювати напрямок пучка.

За інших рівних умов чіткість зображення на моніторі тим вища, що менше розмір точки люмінофора (Dot Pitch) на внутрішній поверхні екрана. Розмір точок, а точніше, середня відстань між ними називається зерном. У різних моделей моніторів цей параметр має значення від 0,21 до 0,41 мм (у добрих моніторів – не більше 0,28 мм).

Недоліки моніторів на основі ЕЛТ:

великі маса та габарити;

значне енергоспоживання; наявність тепловиділення;

випромінювання, шкідливі здоров'ю;

значна нелінійність растру, складність її корекції.

Рідкокристалічні монітори

Основним елементом РК-монітора або інакше LCD-монітора (Liquid Crystal Display) є РК-екран, що складається з двох панелей, виконаних зі скла, між якими розміщений шар рідкокристалічної речовини. Ці скляні панелі зазвичай називають підкладками. Як і у звичайному моніторі, екран РК-монітора є сукупністю окремих елементів – РК-комірок, кожна з яких генерує один піксел зображення. Однак, на відміну від зерна люмінофора ЕПТ, РК-комірка сама не генерує світло, а лише керує інтенсивністю світла, що проходить, тому РК-монітори завжди використовують підсвічування.

По суті РК-осередок є електронно-керованим світлофільтром, принцип дії якого заснований на ефекті поляризації світлової хвилі. Рідкокристалічна речовина, розміщена між підкладками, має молекули витягнутої форми, які називаються нематичними. Завдяки цьому молекули РК-речовини мають упорядковану орієнтацію, що призводить до появи оптичної анізотропії, коли показник заломлення РК-речовини залежить від напрямку поширення світлової хвилі. Якщо нанести на підкладки дрібні борозенки, молекули РК-речовини будуть орієнтовані вздовж цих борозенок. Іншою важливою властивістю РК-речовини є залежність орієнтації молекул від напрямку зовнішнього електричного поля. Використовуючи дві ці властивості, можна створити електронно-керований світлофільтр.

На сьогоднішній день існують такі технології РК-матриц:

- TN та TN+film;
- TFT;
- IPS (SFT);
- PLS;
- VA, MVA, PVA.

Основні характеристики РК-моніторів:

- робочий дозвіл;
- інтерфейс;
- тип матриці;
- яскравість;
- контраст;
- кути огляду;
- час реакції пиксела;
- кількість кольорів, що відображаються.

OLED-дисплей

Як і традиційний LCD-дисплей, OLED-панель складається з безлічі пікселів, які формують картинку. Однак у більш дешевого конкурента є ще й шар з підсвічуванням, тут такого елемента немає. Натомість кожен піксель по суті є органічним світлодіодом, що вміє випускати світло самостійно. Тобто яскравість OLED-екрана регулюється попиксельно. У IPS- або TFT-панелі у будь-якому випадку вся площа висвітлюється підсвічуванням, через що контрастність виходить далекою від ідеалу.

OLED-дисплеї складаються з певної кількості тонких органічних плівок між двома провідниками. Саме подача напруги на провідники змушує екран випромінювати світло. Така конструкція дозволяє ще й без особливих труднощів згинати дисплей - деякі виробники вже показували панелі, здатні скручуватися в трубку.

OLED матриці мають наступні переваги:

- картинка не має спотворень за будь-якого вугілля огляду;
- Відсутність окремого шару з підсвічуванням позитивно позначається на енергоспоживанні;
- такі екрани мають мінімальну товщину, у зв'язку з чим зменшуються фізичні розміри кінцевого пристрою;
- контрастність виходить практично ідеальною, адже при показі чорних кольорів підсвічування практично повністю згасає;
- технологія OLED дозволяє виводити більше кольорів.

Лекція 3 (1 год.) Аудіосистема комп'ютера. Компоненти звукової карти.

План лекції:

1. Класифікація звукових карт.
2. Основні параметри звукових карт.
3. Типові роз'єми звукових карт.

Всі звукові плати, які використовуються в ПК можна розділити на три групи за їх призначенням:

- *звукові*, містять тільки тракт цифрового запису/відтворення. Дозволяють тільки записувати або відтворювати безупинний звуковий потік, на зразок магнітофона. Вся робота по запам'ятовуванню записуваного і підготовці відтвореного потоку покладається на програмне забезпечення; оцифрований звук при цьому в самій платі не зберігається. Деякі звукові плати мають вбудовані сигнальні процесори для обробки звуку в процесі його запису або відтворення.

- *музичні*, містять тільки музичний синтезатор. Такі плати орієнтовані, насамперед, на генерацію відносно коротких музичних звуків по командах від центрального процесора; самі звуки при цьому або створюються параметрично, або відтворюються оцифровки, заздалегідь поміщені в пам'ять синтезатора (ПЗП чи ОЗП). Музичні плати не мають можливості запису звуку і, навіть при наявності ОЗП в синтезаторі, не розраховані на відтворення безупинної звукового потоку, хоча іноді цього можна домогтися за допомогою особливих методів. Деякі музичні плати містять ефект-процесор для обробки створюваного звуку.

- *комбіновані*, або звуко-музичні, з об'єднанням на одній платі цифровим трактом і музичним синтезатором.

У комбінованих картах можна виділити чотири блоки:

- блок цифрового запису / відтворення, який називають також цифровим каналом, або трактом, карти. Здійснює перетворення аналог-цифра і цифра-аналог в режимі програмної передачі або по DMA. Цифровий канал більшості поширених карт (крім GUS) сумісний з Sound Blaster Pro (8 розрядів, 44 кГц – моно, 22 кГц – стерео).

- блок синтезатора. Побудований або на базі мікросхем FM-синтезу, або на базі мікросхем WT-синтезу, або того й іншого разом. Працює або під управлінням драйвера (FM, більшість WT) – програмна реалізація MIDI – або під управлінням власного процесора – апаратна реалізація. Більшість WT-синтезаторів містить вбудований ПЗП зі стандартним набором інструментів General MIDI, а також ОЗП для завантаження додаткових оцифрованих звуків, які будуть використовуватися при виконанні музики.

- блок MPU (MIDI Processing Unit – пристрій MIDI-обробки). Здійснює прийом/передачу даних по зовнішньому MIDI-інтерфейсу, виведеним на роз'єм MIDI/Joystick і роз'єм для дочірніх МШП-плат. Зазвичай більш-менш сумісний з інтерфейсом MPU-401, але найчастіше потрібно програмна підтримка.

- блок мікшера. Здійснює регулювання рівнів, комутацію і зведення використовуваних на карті аналогових сигналів. До складу мікшера входять попередні, проміжні і вихідні підсилювачі звукових сигналів.

У дочірніх платах основними блоками є власне музичний синтезатор та блок MIDI-інтерфейса, через який плата отримує MIDI-повідомлення з основної картки. Синтезатор обов'язково має ПЗП різного обсягу; наявність

ОЗП можливе, але незручно, оскільки MIDI є досить повільним для завантаження оцифровок інтерфейсом. Синтезований звук повертається в основну карту по аналоговому стереоканалу.

Основні параметри звукової карти – розрядність, максимальна частота дискретизації, кількість каналів (моно/стерео), параметри синтезатора, розширенність, сумісність.

Розрядність карти – розрядність цифрового подання звуку. Існують 8-бітові, 16-бітові, 32-бітові та 64-бітові мережеві карти. 8-розрядність карти дають якість звуку, близьку до телефонного; 16-розрядність вже підходять під визначення "Hi-Fi" і теоретично можуть забезпечити студійну якість звучання, хоча практично це реалізується дуже рідко. Розрядність подання звуку не має ніякого зв'язку з розрядністю системної шини для карти, однак карта для 32-розрядної шини MCA, EISA, VLB або PCI буде працювати із трохи меншими накладними витратами на запис/відтворення оцифрованого звуку, ніж карта для ISA.

Максимальна частота дискретизації (оцифровки) визначає максимальну частоту записуваного/відтвореного сигналу, яка приблизно дорівнює половині частоти дискретизації. Для запису / відтворення мови може бути досить 6-8 кГц, для музики середньої якості – 20-25 кГц, для високоякісного звучання необхідно 44 кГц і більше. У деяких картах можна підвищити частоту дискретизації ціною відмови від стереозвуку: два канали по 22 кГц, або один канал на 44 кГц.

Параметри синтезатора визначають можливості карти в синтезі звуку і музики. Тип синтезу – FM чи WT – визначає вид звучання музики: на FM-синтезатор інструменти звучать дуже бідно, зі "дзвінком" відтінком, імітація класичних інструментів дуже умовна; на WT-синтезатор звучання більш "живе", "соковите", класичні інструменти звучать природно, а синтетичні – більш приємно, на хороших WT-синтезаторах може навіть створитися враження "живої гри" чи "слухання CD".

Розширюваність визначає можливості по підключенню додаткових пристроїв, установці мікросхем, розширення об'єму ПЗП чи ОЗП і т.п. На багатьох картах є 26-розрядний внутрішній роз'єм для підключення дочірньої плати, що представляє собою додатковий WT-синтезатор. Практично на кожній карті є роз'єм для підключення CD-ROM для підключення до студійного устаткування, роз'єми для підключення модему та інші. Деякі карти допускають установку DSP і додаткової пам'яті для семплів WT-синтезатора.

Сумісність – зараз найчастіше розуміється сумісність з моделями Sound Blaster – зазвичай SB Pro і SB 16 (для карт виробництва Creative і карт на мікросхемі Creative Vibra 16).

На типовій звуковій карті можуть знаходитися такі *роз'єми*:

1. ігровий, або MIDI-порт. Самий великий і помітний 15-контактний роз'єм-гніздо, призначений для підключення джойстика, MIDI-клавіатури або чогось іншого, працюючого через MIDI-інтерфейс, наприклад, синтезатор.

2. лінійний вхід
3. мікрофонний вхід
4. лінійний вихід для підключення активних колонок або підсилювача.
5. аудіо вихід, на який подається пройшовший через вбудований в карту малопотужний (2-4 вата на канал) підсилювач сигнал.
6. цифровий вихід – він призначений для підключення зовнішніх цифрових пристроїв, наприклад цифрового ресівера. Зустрічається тільки на досить дорогих картах.
7. Цифровий вхід – зустрічається ще рідше, ніж цифровий вихід.

Контрольні запитання:

1. Для чого призначено відеоадаптер?
2. Склад відеоадаптера.
3. Класифікація моніторів.
4. Основні характеристики РК моніторів.
5. Склад звукової кати.
6. Основні характеристики звукових карт.

Тема 3.1 Подання чисел у комп'ютерах. Обробка чисел із плаваючою комою.

Лекція 1 (2 год.) Числа зі знаком. Операції з числами зі знаком.

План лекції:

1. Подання чисел зі знаком.
2. Особливості операції з числами зі знаком.
3. Переповнення в цілісній арифметиці.

Усі комп'ютери працюють із числами. Вони мають команди реалізації базових арифметичних операцій із даними. Крім того, при виконанні машинних команд програми виконується ряд арифметичних операцій, що генерують числові адреси для доступу до операндів, що зберігаються в пам'яті. Щоб зрозуміти, як вирішуються ці завдання, студент повинен знати, як числа представлені в комп'ютері і як вони складаються і віднімаються.

Комп'ютери працюють лише з числовими даними, але й із символами і навіть із рядками символів, тобто поряд із числовою інформацією оперують і текстовою. Тому поряд з числами та операціями над ними буде розглянуто машинне представлення символів.

Комп'ютери складаються з логічних схем, які обробляють інформацію як електричних сигналів, приймають два значення. Ми позначаємо їх цифрами 0 та 1. Кількість інформації, поданої таким сигналом, вимірюється в бітах. Найбільш природний спосіб представлення числа в комп'ютерній системі полягає у використанні рядка бітів, що називається двійковим числом. Символ тексту також може бути представлений рядком бітів, який називається кодом символу.

Розглянемо n -розрядний вектор

$$V = b_{n-1} \dots b_1 b_0$$

Тут $b_i = 0$ або 1 при $0 \leq i \leq n-1$. Цей вектор може представляти беззнакове ціле значення V в діапазоні від 0 до 2^n-1 , де

$$V(V) = b_{n-1} \times 2^{n-1} + \dots + b_1 \times 2^1 + b_0 \times 2^0.$$

Цілком очевидно, що нам потрібно якось представляти і додатні, і від'ємні числа. Існують три системи представлення чисел зі знаком:

- значення зі знаком;
- доповнення до одиниці;
- доповнення до двох.

У всіх трьох системах крайній ліворуч біт, який зветься *найстаршим розрядом* (Most Significant Bit, MSB), дорівнює 0 у разі позитивних чисел і 1 – у разі негативних.

У системі значення зі знаком від'ємні числа відрізняються від відповідних додатних чисел тим, що значення найстаршого біта у векторі дорівнює не 0 , а 1 . Наприклад, число $+5$ представляється як 0101 , а число -5 як 1101 . Таке подання чисел називають ще **прямим кодом числа зі знаком**.

У поданні доповнення до одиниці від'ємні значення отримують шляхом доповнення кожного розряду відповідного додатного значення до одиниці, тобто. щоб зробити число від'ємним, потрібно замінити кожну одиницю нулем і кожен нуль одиницею. Таке подання чисел ще називають **зворотним кодом числа**.

У системі доповнення до двох операція доповнення проводиться шляхом віднімання числа з 2^n . Те саме значення можна отримати і шляхом додавання 1 до доповнення цього числа до одиниці. Таким чином, заперечення числа відбувається у два етапи. Спочатку кожна одиниця змінюється на нуль, а кожен нуль – на одиницю (як і в системі доповнення до одиниці). Потім до отриманого результату додається одиниця. Таке подання числа ще називають **додатковим кодом числа**.

Десятичні числа	Доповнення до одиниці	Доповнення до двох
$\begin{array}{r} 10 \\ + (-3) \\ \hline +7 \end{array}$	$\begin{array}{r} 00001010 \\ 11111100 \\ \hline 1\ 00000110 \\ \text{Перенесення 1} \\ 00000111 \end{array}$	$\begin{array}{r} 00001010 \\ 11111101 \\ \hline 1\ 00000111 \\ \text{Відкидаються} \\ 00000111 \end{array}$

Рисунок 3.1 – Додавання в системах з доповненням до одиниці та з доповненням до двох

Правила складання та віднімання n-розрядних чисел зі знаком у системі доповнення до двох:

1. Для складання двох чисел слід скласти їх n-розрядні подання, ігноруючи сигнал перенесення з позиції старшого розряду (MSB). Сумою буде правильне алгебраїчне значення, що представлене в системі доповнення до двох, якщо це значення лежить в діапазоні від -2^{n-1} до $+2^{n-1} - 1$.

2. Для віднімання чисел X і Y, тобто виконання операції $X - Y$, слід обчислити доповнення числа Y до двох, а потім додати його до числа X з урахуванням правила 1. Результатом буде правильне алгебраїчне значення, що представлене в системі доповнення до двох, якщо це значення лежить у діапазоні від -2^{n-1} до $+2^{n-1} - 1$.

Правила переповнення при операціях з числами зі знаком у системі доповнення до двох:

1. Переповнення може відбутися лише при додаванні чисел з однаковими знаками.

2. При додаванні чисел зі знаком сигнал перенесення з позиції знакового біта не є індикатором переповнення.

Лекція 2 (2 год.) Стандарт числа зі плаваючою комою. Особливості операції із числами зі плаваючою комою.

План лекції:

1. Формат числа із плаваючою комою.

2. Стандарт IEEE для чисел із плаваючою комою.
3. Арифметичні операції з числами із плаваючою комою.

Розглянемо діапазон значень, які можна подати у 32-розрядному форматі з фіксованою комою. Якщо інтерпретувати їх як цілі числа, виходить діапазон від 0 до $\pm 2,15 \cdot 10^9$, а якщо як дробі – діапазон від $\pm 4,55 \cdot 10^{-10}$ до ± 1 . Жодного з цих діапазонів недостатньо для наукових обчислень, у яких можуть використовуватися такі параметри, як число Авогадро ($6,0247 \cdot 10^{23}$ моль⁻¹) та константа Планка ($6,6254 \cdot 10^{-34}$ Дж·с). Необхідний такий формат, який підходив би і для дуже великих цілих, і для маленьких дробових чисел. Для цього комп'ютер повинен вміти представляти числа та оперувати ними таким чином, щоб позиція двійкової коми була змінною та автоматично змінювалась у процесі обчислень. Такі числа називають **числами з плаваючою комою**. Нагадаємо, що в числах з фіксованою комою двійкова кома завжди розташовується в одній і тій же позиції.

Визначення машинного подання числа з плаваючою комою: *воно складається зі знака, рядка значущих цифр, званої мантисою, і показника ступеня (фіксованої основи), званого порядком*.

Спочатку розглянемо у загальних рисах уявлення чисел з плаваючою комою у десятковій системі числення, та був співвіднесемо це уявлення з двійковим. Отже, числа з плаваючою комою зручно записувати так:

$$\pm X_1, X_2, X_3, X_4, X_5, X_6, X_7, \times 10^{\pm Y_1 Y_2}$$

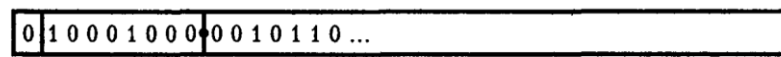
Тут X_i і Y_i – це десяткові цифри. Такої кількості цифр (7) і діапазону значень порядку (± 99) достатньо для наукових розрахунків. Мантису і порядок з цих діапазонів можна перевести в двійкову виставу, що вміщується в 32 розряди, тобто стандартне комп'ютерне слово. Мантиса довжиною 24 біта може представляти десяткове число з 7 цифр, а 8-бітовий порядок основи 2 дозволяє представити досить широкий діапазон значень масштабних множників. Для знаку числа потрібний один біт. Оскільки провідний біт нормалізованої мантиси обов'язково має дорівнювати 1, його можна не включати в уявлення, за рахунок чого і звільняється один біт для знака. Таким чином, 32 біти дозволяють уявити досить широкий діапазон чисел з плаваючою комою.

Описаний стандарт представлення чисел з плаваючою комою у 32-розрядному форматі розроблено та детально специфіковано Інститутом інженерів з електротехніки та електроніки (Institute of Electrical and Electronics Engineers, IEEE). У ньому визначено і подання чисел, та правила виконання чотирьох базових арифметичних операцій.

Знак числа задається в першому розряді, за ним слідує уявлення порядку (на підставі 2). Замість числа зі знаком у полі порядку E зберігається ціле число без знака $E' = E + 127$. Цей формат називається форматом з надлишком 127. Таким чином, E' входить у діапазон $0 \leq E' \leq 255$. Граничні значення зазначеного діапазону, 0 і 255, використовуються уявлення описаних нижче спеціальних значень. Для звичайних (нормальних) значень E' лежить у межах $1 \leq E' \leq 254$.

Це означає, що реальний порядок E знаходиться в діапазоні $-126 \leq E \leq 127$. Подання порядку у форматі з надлишком x спрощує порівняння відносного розміру двох чисел з плаваючою комою.

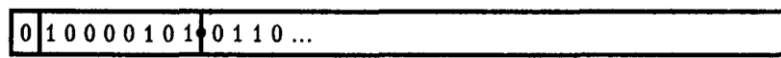
Останні 23 розряди числа представляють мантису. Оскільки числа задаються нормалізованому вигляді, старший біт мантиси завжди дорівнює 1. Цей біт не вказується явно; мається на увазі, що він розташований ліворуч від двійкової коми. Таким чином, 23 біти в полі M відповідають дробовій частині мантиси, тобто розрядам праворуч від двійкової коми.



(Зліва від двійкової коми немає явно заданої 1)

Представлене значення = $+0,0010110... \times 2^9$

а)



Представлене значення = $+1,0110... \times 2^6$

б)

Рисунок 3.2 – Число з плаваючою комою у форматі IEEE одинарної точності: ненормалізоване (а); нормалізоване (б)

Стандартне 32-розрядне подання називається поданням з одинарною точністю, оскільки займає одне 32-розрядне слово. Воно дозволяє представляти масштабні множники в діапазоні від 2^{-126} до 2^{+127} , що дорівнює $10^{\pm 38}$. Для досягнення більшої точності та збільшення діапазону чисел із плаваючою комою стандарт IEEE визначає формат подвійної точності. У ньому розширено діапазони значень мантиси та порядку. Так, 11-розрядний порядок у форматі з надлишком 1023 E' лежить в діапазоні $1 \leq E' \leq 2046$ для звичайних значень, а значення 0 і 2047 використовуються для представлення спеціальних символів. Отже, дійсний порядок E лежить у діапазоні $-1022 \leq E \leq 1023$, що дозволяє уявити масштабні множники від 2^{-1022} до 2^{1023} . Значення 53-розрядної мантиси забезпечують практично таку ж точність, як і 16 десяткових цифр.

Граничні значення 0 та 255 порядку E' у форматі з надлишком 127 використовуються для представлення **спеціальних значень**. Якщо $E' = 0$ і дрібна частина мантиси M дорівнює нулю, отже, представлено точне значення 0. Порядок $E' = 255$ і мантиса $M = 0$ становлять значення ∞ , де ∞ — результат розподілу нормального числа на нуль. Для представлення цих значень зазвичай застосовується і знаковий розряд, наприклад: ± 0 і $\pm \infty$.

Значення $E' = 0$ та $M \neq 0$ відповідають уявленню аномальних чисел. Це числа $\pm 0, M \times 2^{-126}$, які менші за найменше число. У них відсутня мається на увазі одиниця зліва від двійкової коми, а M являє собою будь-яку ненульову 23-розрядну дробову частину числа. Аномальні числа призначені для випадків, коли можлива поступова втрата значущості; вони розширюють діапазон чисел і можуть бути корисні при роботі з дуже маленькими числами. Коли $E' = 255$ і $M \neq 0$, представлене значення називається Not a Number (NaN).

Значення NaN є результатом виконання неприпустимої операції, такої як 0/0 або $\sqrt{-1}$.

Правило додавання та віднімання

1. Вибрати число з меншим порядком і зрушити його мантису вправо на кількість розрядів, що дорівнює різниці порядків.
2. Встановити порядок результату рівним більшому порядку операндів.
3. Виконати додавання/віднімання мантис і визначити знак результату.
4. Нормалізувати результат у разі потреби.

Множення і ділення чисел з плаваючою комою навіть простіше їх складання та віднімання — для виконання цих операцій вирівнювати мантиси не потрібно.

Правило множення

1. Скласти порядки операндів і відняти від результату значення 127.
2. Перемножити мантиси та визначити знак результату.
3. У разі потреби нормалізувати результат.

Правило ділення

1. Відняти порядок дільника з порядку ділимого та додати до результату значення 127.
2. Розділити мантиси та визначити знак результату.
3. У разі потреби нормалізувати результат.

Додавання або віднімання 127 при множенні та діленні виконується тому, що порядки чисел представлені у форматі з надлишком 127.

Контрольні запитання:

1. Які форми подання чисел зі знаком існують?
2. Правила виконання арифметичних операцій з числами у доповненому коді.
3. Коли при операціях з числами у доповненому коді виникає переповнення?
4. Як виглядає формат числа із плаваючою комою?
5. Яке число із плаваючою комою є нормалізованим, а яке є ненормалізованим?
6. Правила виконання арифметичних операцій з числами у форматі із плаваючою комою.

Тема 3.2 Сегментована пам'ять і регістрова структура мікропроцесора i8086.

Лекція 1 (2 год.) Поняття сегмента. Спосіб адресації комірок сегментованої пам'яті.

План лекції:

1. Поняття сегмента пам'яті.
2. Адресація комірок сегментованої пам'яті.
3. Формування фізичної адреси.

Як відомо, МП виконує лише ту програму, яку завантажено в електронну (оперативну) пам'ять. Пам'ять складається з комірок, кожна комірка має свій унікальний номер або адресу. Програміст надає адресі ім'я, а програма-транслятор замінює ім'я на двійковий код. Розрядність шини адреси (ША) МП визначає допустиму кількість (простір) адрес.

Пам'ять МП i8086 організована досить незвично, що пояснюється поєднанням 1-мегабайтної пам'яті та 16-розрядних регістрів. У пам'яті ємністю 1 Мбайт для подання адреси потрібно 20 біт. Отже, зберегти покажчик на пам'ять в одному 16-розрядному регістрі неможливо. З метою вирішення проблеми пам'ять поділена на блоки або **сегменти** по 64 Кбайти, і адреси в рамках цих сегментів вміщуються в 16 біт.

Створюючи програму, програміст організовує – визначає – початок і кінець її сегментів і призначення комірок пам'яті у них. Для отримання програми, що виконується в ехе-форматі, необхідно визначити 3 наступні головні сегменти, а при необхідності – 1 додатковий:

1. *Сегмент кодів.* Містить комірки пам'яті, що зберігають програму, яка виконується (усю сукупність команд у машинних кодах). Перший байт першої команди знаходиться в першій комірці сегмента і т.і.

2. *Сегмент даних.* Містить комірки з даними програми: константами та робочими областями, зарезервованими для запису в них програмою, даних, що вводяться, наприклад, з клавіатури, а також для результатів, що обчислюються програмою. Дані, що оброблюються МП, часто називають операндами, т.я. над ними виконується закодована у команді операція.

3. *Сегмент стека.* Містить комірки, запис та зчитування яких виконується за особливим алгоритмом, відмінним від інших сегментів. Використовується для тимчасового зберігання команд (кодів), даних та проміжних результатів.

4. *Додатковий сегмент.* Містить комірки для зберігання даних та результатів, додаткових до сегмента даних.

Сегменти розміщуються у пам'яті у порядку, у якому їх було визначено програмістом. Програміст надає сегменту ім'я, а транслятор замінює ім'я двійковим кодом.

З урахуванням сегментної адресації пам'яті, адреса будь-якої комірки пам'яті визначається МП додаванням двох складових (логічних адрес).

Перша складова – це *адреса сегмента* $A_{\text{сегм}}$, в якому знаходиться шукана комірка. Друга – це адреса комірки, що відраховується від початку цього сегмента, і називається тому *адресою зміщення комірки в сегменті* $A_{\text{зміщ}}$.

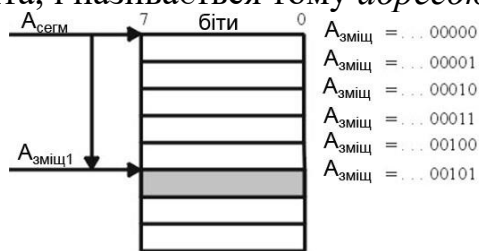


Рисунок 3.3 – Розміщення в пам'яті байта

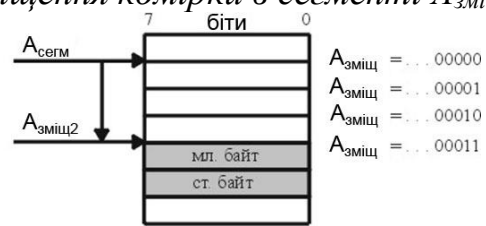


Рисунок 3.4 – Розміщення в пам'яті слова

Пам'ять має байтову організацію – двобайтове слово розміщується у суміжних комірках, причому старший байт займає комірку з великим номером. Адресою слова є адреса молодшого байта.

Такий спосіб адресації комірок пам'яті зручний тим, що при зміні $A_{\text{сегм}}$ не потрібно змінювати $A_{\text{зміщ}}$: значення адрес зміщень завжди відраховуються від початку сегмента і не залежать від значення адреси сегмента.

Таким чином, у МП повинні зберігатися адреси всіх сегментів програми, що виконується. При кожному зверненні до пам'яті з метою прочитати команду або операнд або записати результат, у МП має бути сформована **виконавча адреса** необхідної комірки як доданок $A_{\text{сегм}}$ та $A_{\text{зміщ}}$, який МП вибирає з команди або обчислює за даними команди.

Хоча всі регістри зміщень та сегментів є 16-розрядними, МП видає на шину адреси при зверненнях до оперативної пам'яті (ОП) 20-розрядні виконавчі (фізичні) адреси, що дозволяють звертатися до ОП ємністю 1 Мбайт. Це стає можливим завдяки механізму сегментації пам'яті. Розмір сегмента не може перевищувати 64 Кбайти. Допускається перекриття сегментів. Базові (початкові) 16-розрядні адреси сегментів, що зберігаються у відповідних сегментних регістрах, трактуються як 20-розрядні з нулями у чотирьох молодших розрядах. Іншими словами, початкові адреси мають бути кратними 16.

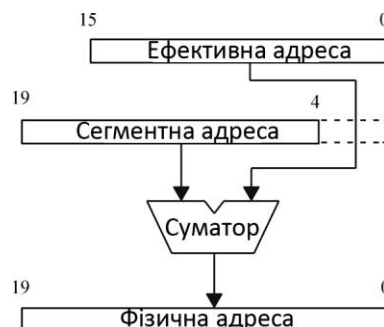


Рисунок 3.5 – Формування фізичної адреси

Фізична адреса операнда чи команди формується, як показано на рис. 3.5, додаванням початкової адреси відповідного сегмента (вмісту сегментного регістру, зсунутого на чотири розряди вліво, тобто доповненого нулями в чотирьох молодших розрядах) та адреси зміщення, так само званого

ефективною адресою ЕА. Адреса зміщення обчислюється в МП на основі інформації про спосіб формування адреси операнда, що міститься в команді.

Лекція 2 (2 год.) Програмістська модель МП i8086. Призначення регістрів.

План лекції:

1. Загальні уявлення о програмістській моделі МП i8086.
2. Регістри загального призначення.
3. Сегментні та вказівні регістри.
4. Регістр прапорів.

Сукупність регістрів, доступних програмісту і використовуваних ним під час складання програм мовою Ассемблер чи іншій машинно-орієнтованій мові, називається *програмістською моделлю*.

На рис. 4 представлено програмістську модель МП i8086, що містить програмно-доступні регістри. До них відносяться регістри загального призначення, сегментні регістри, регістр - покажчик команд і регістр ознак - прапорів. Усі вони є 16-розрядними регістрами.



Рисунок 3.6 – Програмістська модель МП i8086

Регістри загального призначення – це регістри, які можна використовувати у будь-яких арифметичних, логічних та інших машинних операціях.

Регістри даних AX, BX, CX, DX служать для зберігання операндів та результатів операцій. Можлива адресація як цілих регістрів, і їх молодшої і старшої частин, які зветься L і H відповідно. Деяким регістрам поряд із

загальним призначенням надаються й спеціальні. У останньому випадку у відповідних командах ці регістри вказуються неявно самим кодом операції. Так, у деяких командах

- регістр AX використовується як акумулятор, тобто. регістра, в якому знаходиться один з операндів і який міститься результат операції;
- регістр BX – як базовий регістр, що використовується при непрямій адресації;
- регістр CX – як лічильник кількості зміщених біт у командах зсуву, лічильник числа повторів у командах управління обчислювальними циклами та операціях з ланцюжками байт;
- регістр DX неявно адресується в командах множення та поділу, а в деяких операціях введення-виведення зберігає адресу порту введення-виводу.

МП i8086 має для зберігання адрес сегментів поточної програми 4 *сегментні регістри*: CS - для сегмента коду, DS – для сегмента даних, SS – для сегмента стека, ES – для додаткового сегмента. У жодних арифметичних, логічних і т.і. машинних операціях ці регістри брати участь неспроможні. Можна лише записувати в них і зчитувати з них та й то є певні обмеження.

Групу *вказівних та індексних регістрів*, що задають внутрішньосегментні зміщення, утворюють такі регістри: покажчика стека (SP), покажчика бази стека (BP), індексу операнда (джерела) (SI) та індексу результату (приймача) (DI). До цієї групи можна віднести і регістр покажчика (лічильника) команд – IP, що часто називається лічильником команд. Причому сегментні, індексні та вказівні регістри застосовуються попарно, відповідно до сегментної адресації таким чином:

CS та IP – адресують разом вміст сегмента коду програми;

SS і BP або SS та SP – адресують сегмент стека;

DS та один з регістрів BX, SI, DI – адресують сегмент даних;

Регістр ознак (прапорів) має 16 розрядів, причому молодші 8 розрядів відповідають регістру прапорів МП i8080. У регістрі ознак зберігаються:

- сформовані в АЛУ такі ознаки результату виконаної операції:
- переповнення OF (при операціях з цілими числами);
- знака SF, якщо результат негативний – SF=1;
- нуля ZF якщо результат нульовий – ZF=1;
- допоміжного перенесення AF (перенесення з третього або позики до третього розряду);
- паритету PF (PF=1, якщо число одиниць у молодшому байті результату непарне);
- перенесення CF (CF=1, якщо було перенесення зі старшого або позику до старшого розряду результату);
- ознаки керування, що встановлюються пристроєм керування:
- покрокового режиму TF (керує покроковими перериваннями)
- дозволу переривання IF (дозволяє або забороняє переривання, що маскуються);

- напрям DF (вказує напрям обробки ланцюжка даних, починаючи з елемента з найменшою адресою при DF=0 або найбільшою адресою при DF=1).

Лекція 3 (2 год.) Способи адресації даних у командах мови Асемблер.

План лекції:

1. Регістрова адресація.
2. Безпосередня адресація.
3. Пряма та непряма адресація.
4. Адресація по базі та індексна адресація.
5. Адресація по базі з індексуванням.

У загальному випадку дані (операнди і результат) можуть зберігатися у пам'яті, портах, регістрах. У програмах на мові Асемблер програміст сам розподіляє пам'ять та регістри під операнди та результати. Місце розташування (інакше – адреса) програміст вказує у кожній команді, дотримуючись правил синтаксису мови.

Головні правила синтаксису полягають у наступному:

1. В одній команді тільки один з операндів може бути розміщений у пам'яті.
2. Регістр може зберігати операнд або адресу (зміщення), яку має бути заключено у квадратні дужки.
3. Розмір операнда визначається розміром регістру.
Наприклад: AX – зберігає лише слово;
BH або AL – лише байт.

4. Результат операції за замовчуванням зберігається за адресою першого операнда (за замовчуванням – означає, що відомості у команді про це відсутні, тому транслятор приймає певний варіант – адреса першого операнда).

Регістрова адресація має місце, якщо в команді вказується ім'я регістру, при цьому враховується розмір операнда:

– якщо в команді вказано, наприклад, AL, то операнд зберігається в регістрі AL та його розмір дорівнює одному байту. Те саме для всіх одnobайтних регістрів.

– якщо в команді вказано, наприклад, BX, то операнд зберігається в BX і має розмір слова (два байти). Те саме для всіх двобайтних регістрів.

Для зберігання операндів та результатів можуть використовуватися регістри загально призначення (РЗП), індексні регістри та регістри-показники.

Приклад:

```
MOV AX, BX    ; AX:=BX
ADD CX, AX    ; CX:=AX
PUSH CX       ; Заштовхнути в стек вміст CX
```

Безпосередня адресація. Операнд (8- чи 16-розрядна константа) міститься у тілі команди, тобто безпосередня адресація має місце, якщо операнд вказується в команді чисельним значенням (константа) у будь-якій допустимій системі числення (2, 10, 16). Так як результат за замовчанням зберігається у першому операнді, не можна адресувати безпосередньо операнд, розміщений першим (тобто. не можна MOV 15, AX).

Приклад:

```
MOV AX, 100          ; AX:=100
ADD AX, 5             ; AX:=AX+5
MOV CX, 0FFFFH       ; CX:=65535
```

Пряма адресація. Пряма адресація має місце, якщо операнд вказується ім'ям, визначеним в одному із сегментів.

Наприклад:

```
MOV AX, X; AX:=X      ( Слово! )
ADD AH, B; AH:=AH+B   ( Байт! )
; де X і B раніше описані
```

Непряма реєстрова адресація. Виконавча адреса операнда (точніше його зміщення) міститься в одному з реєстрів BX, BP, SI або DI. Для вказівки непрямої адресації цей реєстр повинен полягати у дужках:

```
MOV AX, [BX] ; AX:= слово, що міститься за адресою DS:BX
```

Кожен із реєстрів за замовчуванням працює зі своїм сегментним реєстром: DS:BX; SS:BP; DS:SI; ES:DI. Дозволяється явна вказівка сегментного реєстру, якщо він відрізняється від прийнятого за умовчанням:

```
MOV AX, ES:[BX]
```

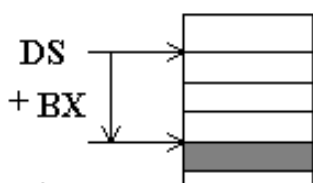


Рисунок 3.7 – Непряма реєстрова адресація.

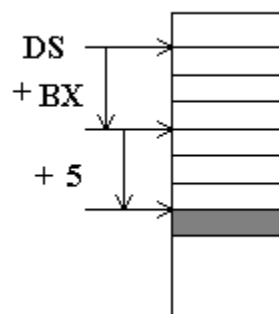


Рисунок 3.8 – Адресація по базі.

Адресація по базі. Базовий реєстр BX або BP містить базу (адреса початку деякого сегмента пам'яті), щодо якої асемблер обчислює зміщення.

Приклад:

```
MOV AX, [BX]+5
```

Індексна адресація. Один з індексних реєстрів SI або DI вказують положення елемента щодо початку деякої області пам'яті. Нехай, наприклад, АОВ – масив байтів. Тоді:

MOV SI, 15 ; SI:=15
MOV AH, AOB[SI] ; AH:=[AOB+15], тобто 16-й по порядку
 ; байт від початку масиву
MOV SI, 0 ; SI:=0
MOV AOB[SI], AH ; пересилаємо AH у перший елемент масиву

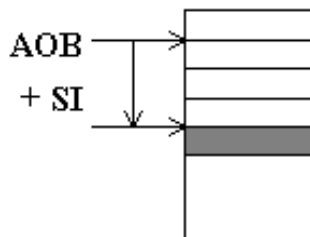


Рисунок 3.9 – Індексна адресація.

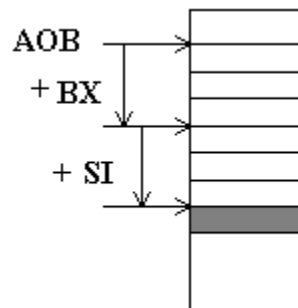


Рисунок 3.10 – Адресація по базі з індексуванням.

Адресація по базі з індексуванням. Варіант індексної адресації для випадку, коли область пам'яті, що індексується, задається своєю базою.

Приклад:

MOV AX, [BX][SI]

Цей тип індексації зручний при обробці двовимірних масивів. Якщо, наприклад, AOB розглядаємо як масив 10x10 байт, для доступу до елемента AOB[2, 3] можна використовувати фрагмент:

MOV BX, 20 ; BX:=20, база строки 2
MOV SI, 2 ; SI:=2, номер 3-го елемента
MOV AX, AOB[BX][SI] ; доступ до елемента

Контрольні запитання:

- З чого складається адреса будь-якої комірки сегментованої пам'яті?
- Який може бути розмір сегменту для 16-розрядного МП?
- Який регістр вказує на сегмент, що містить команди (код) поточної (виконуваної) програми?
- Який із регістрів вказує на сегмент, що містить стек для поточної (виконуваної) програми?
- Вкажіть правильний запис команди за безпосередньої адресації другого операнда:
 - 1) MOV AX, CX
 - 2) MOV AX, [BX]
 - 3) MOV AL, 10
 - 4) MOV AX, TABLE
- Вкажіть правильний запис команди для непрямої регістрової адресації другого операнда:
 - 1) MOV AX, CX
 - 2) MOV AX, [BX]
 - 3) MOV AL, 10
 - 4) MOV AX, TABLE

Тема 3.3 Система команд процесорів.

Лекція 1 (2 год.) Команди пересилання даних та арифметичні команди процесора.

План лекції:

1. Функції команд пересилання даних.
2. Основні мнемоніки команд пересилання даних.
3. Функції арифметичних команд.
4. Команди операцій із фіксованою та плаваючою комою.
5. Команда інкременту-декременту та порівняння.

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

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

У різних процесорів системи команд суттєво відрізняються, але в основі вони дуже схожі. Кількість команд у процесорів також різна. У сучасних потужних процесорів кількість команд досягає декілька сотень.

Команди пересилання даних виконують такі найважливіші функції:

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

У деяких процесорах ці функції виконуються однією єдиною командою **MOV** (для байтових пересилок - **MOVB**).

В інших процесорах, крім цієї команди, для завантаження регістрів можуть використовуватися команди завантаження на основі **LOAD**. Часто виділяються спеціальні команди для збереження в стеку та для вилучення зі стека (**PUSH** – зберегти у стеку, **POP** – витягти зі стека).

Іноді в систему команд вводиться спеціальна команда **MOVS** для малого (або ланцюгового) пересилання даних (і8086). Ця команда пересилає задану кількість слів або байтів (**MOSB**).

У деяких процесорах (і8086) спеціально виділяються функції обміну із пристроями вводу/виводу.

Команда **IN** використовується для введення інформації із пристрою вводу/виводу, а команда **OUT** – для виведення у пристрій введення/виводу. До команд пересилання даних також належать команди обміну інформацією (з урахуванням слова Exchange).

Арифметичні команди можуть бути поділені на п'ять основних груп:

- команди операцій з фіксованою комою (додавання, віднімання, множення, ділення);
- команди операцій із плаваючою комою;

- команди очищення;
- команди інкременту та декременту;
- команди порівняння.

Команди операцій із фіксованою комою працюють із кодами в регістрах процесора чи пам'яті як із звичайними двійковими кодами. Команда додавання (**ADD**) обчислює суму двох кодів. Команда віднімання (**SUB**) обчислює різницю двох кодів. Команда множення (**MUL**) обчислює добуток двох кодів. Команда розподілу (**DIV**) обчислює окреме від розподілу одного коду на інший.

Команди операцій з плаваючою комою використовують формат представлення чисел з порядком та мантиєю (зазвичай ці числа займають дві послідовні комірки пам'яті).

*Команди очищення (**CLR**)* призначені для запису нульового коду в регістр або комірку пам'яті.

Команда інкременту (збільшення на одиницю, **INC**) та *декременту* (зменшення на одиницю, **DEC**) також бувають дуже зручними. Ці команди вимагають одного вхідного операнда, який одночасно є вихідним операндом.

*Команда порівняння (**CMR**)* призначена для порівняння двох вхідних операндів. Вона обчислює різницю двох операндів, але вихідного операнда не формує, а лише змінює біти в регістрі стану процесора за результатом цього обчислення.

Лекція 2 (2 год.) Логічні команди. Команди переходу.

План лекції:

1. Основні операції, що виконуються логічними командами.
2. Мнемоніка логічних команд.
3. Функції команд переходу.
4. Мнемоніка команд переходу.

Логічні команди розглядають коди операндів ні як єдине число, а як набір окремих бітів. Цим вони відрізняються від арифметичних команд.

Логічні команди виконують такі основні операції:

- логічне І, логічне АБО, додавання за модулем 2 (АБО-НЕ);
- логічні, арифметичні та циклічні зрушення;
- перевірка бітів та операндів;
- установка та очищення бітів (прапорів) регістру стану процесора (PSW).

Команди логічних операцій дозволяють бітно обчислювати основні логічні функції від двох вхідних операндів. Операція І (**AND**) використовується для примусової очистки заданих бітів. Операція АБО (**OR**) застосовується для примусового встановлення заданих бітів. Операція АБО-НЕ (**XOR**) використовується для інверсії заданих бітів.

Команди зсувів дозволяють бітно зрушувати код операнда праворуч (у бік молодших розрядів) або вліво (у бік старших розрядів). Тип зсуву

(логічний, арифметичний чи циклічний) визначає значення старшого чи молодшого біта.

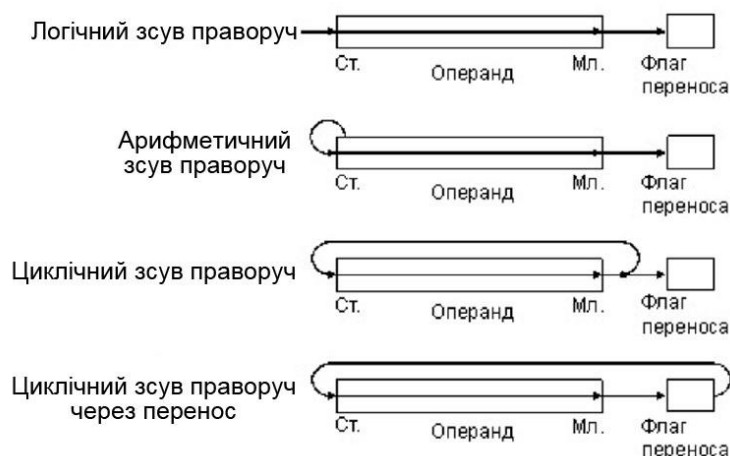


Рисунок 3.11 – Команди зсуву праворуч.

Команди перевірки бітів та операндів призначені для встановлення або очищення бітів регістру стану процесора залежно від значення вибраних бітів або всього операнда загалом. Вихідного операнда команди не формують.

Команди установки та очищення бітів регістру стану процесора дозволяє встановити чи очистити будь-який прапор. Кожен прапор зазвичай відповідає дві команди, одна з яких встановлює його в одиницю, а інша скидає в нуль.

Команди переходів призначені для організації різноманітних циклів, розгалужень, викликів підпрограм тощо, тобто вони порушують послідовний хід виконання програми.

Команди переходу без повернення поділяються на дві групи:

- команди безумовного переходу;
- команди умовного переходу

В позначення цих команд використовуються слова **Branch** і **Jump**.

Команди безумовних переходів викликають перехід у нову адресу незалежно від чого.

Команди умовних переходів викликають перехід який завжди, лише за виконання заданих умов. Як такі умови зазвичай виступають значення прапорів у регістрі стану процесора.

*Команди переходів з подальшим поверненням у точку, з якої було здійснено перехід, застосовується до виконання підпрограм, тобто допоміжних програм (поширена назва - **CALL**).*

Для повернення до точки виклику підпрограми (точку переходу) використовується спеціальна команда повернення (**RET** або **RTS**).

Команди переривань (**INT**) як вхідний операнд вимагають номер переривання.

Для повернення з підпрограми, викликаної командою переривання, використовується команда повернення з переривання (**IRET** або **RTI**).

Контрольні запитання:

1. На які групи ділиться система команд будь-якого процесора?
2. Які функції виконують команди пересилання даних?
3. Які функції виконують арифметичні команди?
4. У чому принципова різниця між арифметичними та логічними командами?
5. Чим відрізняються команди логічного та арифметичного зсувів?

Тема 3.4 Організація взаємодії процесора з основною пам'яттю через кеш пам'ять. Віртуальна пам'ять.

Лекція 1 (2 год.) Загальні принципи функціонування кеш пам'яті. Порядок взаємодії процесора та основної пам'яті через кеш-пам'ять..

План лекції:

1. Загальні функції кеша.
2. Структура системи з основною і кеш пам'яттю.
3. Стратегії оновлення вмісту кеша.

Кеш-пам'ять персональних комп'ютерів є високошвидкісним буфером, побудованим на мікросхемах SRAM (Static RAM – статична оперативна пам'ять), який безпосередньо обмінюється даними з процесором.

У загальному вигляді використання кеш-пам'яті можливо пояснити таким чином. Коли ЦП намагається прочитати слово з основної пам'яті, спочатку здійснюється пошук копії цього слова в кеші. Якщо така копія існує, звернення до ОП не проводиться, а в ЦП передається слово, що є вибраним з кеш-пам'яті. Дану ситуацію прийнято називати успішним зверненням або *попаданням* (hit). За відсутності слова в кеші, тобто у разі неуспішного звернення – *промаху* (miss), – необхідне слово передається в ЦП з основної пам'яті, але одночасно з ОП в кеш-пам'ять пересилається блок даних, що містить це слово.

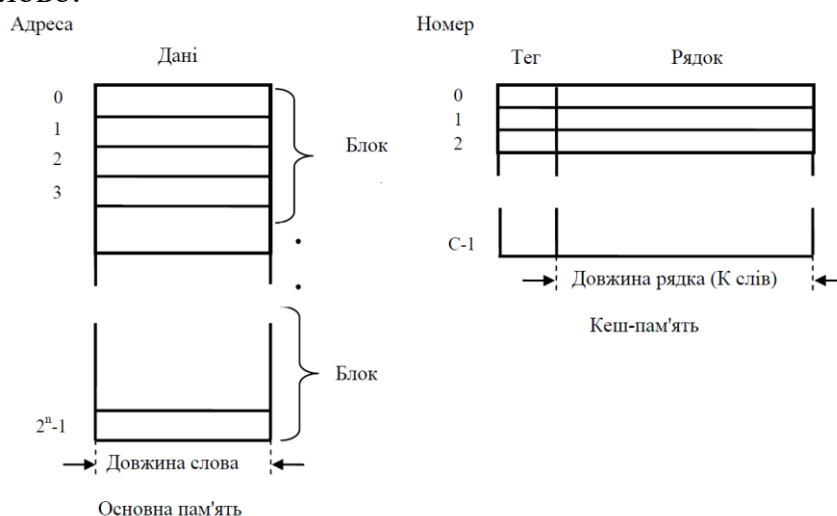


Рисунок 3.12 – Структура системи з основною і кеш пам'яттю.

ОП складається з 2^n слів, що адресуються, де кожне слово має унікальну n -розрядну адресу. Під час взаємодії з кешем ця пам'ять розглядається як M блоків фіксованої довжини по K слів у кожному ($M = 2^n/K$). Кеш-пам'ять складається із C блоків аналогічного розміру (блоки в кеш-пам'яті прийнято називати рядками), причому їх число значно менше числа блоків у основній пам'яті ($C \ll M$). У разі зчитування слова з якого-небудь блока ОП цей блок копіюється в один з рядків кеша. Оскільки число блоків ОП більше числа рядків, окремий рядок не може бути постійно виділений одному і тому ж блоку

ОП. З цієї причини кожному рядку кеш-пам'яті відповідає тег (ознака), що містить відомості про те, копія якого блока ОП у даний момент зберігається в даному рядку.

Інформація про те, який саме блок займає даний рядок, і про його стан зберігається у пов'язаній з даним рядком комірці спеціальної пам'яті тегів (tag RAM). Для процесорів i486 і старших процесорів P5 довжина рядка збігається з об'ємом даних, що передаються за один пакетний цикл (для i486 це – $4 \times 4 = 16$ байт, для Pentium – $4 \times 8 = 32$ байт). Можливий також варіант секторованого (sectored) кеша, в якому один рядок містить кілька суміжних комірок – секторів. Їх розміри відповідають мінімальній порції обміну даних кеша з оперативною пам'яттю. При цьому в записі каталогу, що відповідає кожному рядку, повинні зберігатися біти дійсності для кожного сектора даного рядка. Секторування економить пам'ять, яка необхідна для зберігання каталогу у разі збільшення ємності кеша, оскільки більша кількість бітів каталогу відводиться під тег.

На ефективність застосування кеш-пам'яті в ієрархічній системі пам'яті впливає цілий ряд моментів. До найбільш істотних з них можна віднести:

- ємність кеш-пам'яті;
- розмір рядка;
- спосіб відображення основної пам'яті на кеш-пам'ять;
- алгоритм заміщення інформації в заповнений кеш-пам'ять;
- алгоритм узгодження вмісту основної і кеш-пам'яті;
- число рівнів кеш-пам'яті.

Кеш-пам'ять в сучасних комп'ютерах будується за дворівневою схемою, але може бути і трирівневою.

Первинний (внутрішній) кеш (L1 Cache), або кеш першого рівня, вбудований у процесори, починаючи з i486. Ємність такого кеша порівняно з основною оперативною пам'яттю невелика і становить 8-64 Кбайт, швидкодія – 10-12 нс.

Вторинний (зовнішній) кеш (L2 Cache), або кеш другого рівня, встановлюється на системній платі, а в процесорах P6 – на картриджі в одній упаковці з ядром і підключається до спеціальної внутрішньої шини процесора. Виготовляється у вигляді окремої мікросхеми. Якщо він встановлюється на системній платі, то працює на її частоті й знаходиться поруч з мікросхемою процесора.

Кеш третього рівня (L3 Cache) застосовується не часто, тому що його реалізація багато коштує (ємність повинна бути більша, ніж у вторинного). Вбудований кеш L3 є у процесорів Xeon MP, його розмір досягає 8 Мбайт.

У процесі обчислень ЦП може не тільки зчитувати існуючу інформацію, але і записувати нову, що приводить до оновлення вмісту кеш-пам'яті.

Поведінка кеш-контролера під час здійснення операції запису в пам'ять, коли копія ділянки, що вимагається, знаходиться в певному рядку кеша, визначається його алгоритмом роботи чи стратегією запису (Write Policy).

Існують *дві основні стратегії запису даних з кеша в основну пам'ять: наскрізний запис – WT (Write Through) і зворотний запис – WB (Write Back).*

WT – запис передбачає виконання кожної операції запису, яка потрапляє в кешований блок, одночасно в рядок кеша і в основну пам'ять. При цьому процесор вимушений щоразу здійснювати тривалу операцію запису в основну пам'ять. За такого запису достатньо тільки інформації тега. Але така простота запису не дає високої ефективності. Існують варіанти алгоритму WT із застосуванням *відкладеного буферизованого запису*, за якого дані в основну пам'ять переписуються через FIFO – буфер (First-In First-Out – “першим прийшов – першим і вийшов”) під час вільних тактів шини. Під час використання буферизації процесор повністю звільняється від роботи з основною пам'яттю.

WB – запис дає змогу зменшити кількість операцій запису на системній шині основної пам'яті. Коли блок основної пам'яті, в який повинен здійснюватися запис, відображений в кеші, то фізичний запис спочатку відбуватиметься в цей дійсний рядок кеша і буде позначений як нечистий (dirty) або модифікований, тобто такий, що вимагає вивантаження в основну пам'ять. Тільки після цього вивантаження рядок стане чистим (clean), і його можна буде використати для кешування інших блоків без втрати цілісності даних. В основну пам'ять дані переписуються тільки цілим рядком. WB-алгоритм складніший в реалізації, ніж WT-алгоритм, але істотно ефективніший. Підтримка кешування із зворотним записом потребує додаткових інтерфейсних сигналів, якщо здійснюється звернення з боку інших контролерів системної шини.

Лекція 2 (2 год.) Основні архітектури кеш пам'яті.

План лекції:

1. Кеш прямого відображення.
2. Повністю асоціативний кеш.
3. Набірно-асоціативний кеш.

Залежно від способу визначення взаємної відповідальності рядка кеша і ділянки основної пам'яті розрізняють три архітектури кеш-пам'яті: *кеш прямого відображення* (direct-mapped cache), *повністю асоціативний кеш* (fully associative cache) та їх комбінація – *набірно-асоціативний кеш* (set-associative cache), який має дві модифікації – множинно-асоціативне відображення і відображення секторів.

Кеш прямого відображення. Адреса пам'яті, за якою відбувається звернення до кеша прямого відображення, однозначно визначає рядок кеша, де може знаходитись необхідний блок.

Сутність архітектури прямого відображення полягає в тому, що кожен рядок кеша може відображати з будь-якої сторінки кешованої пам'яті тільки відповідний йому рядок. При цьому на кожен рядок кеша може претендувати багато сторінок пам'яті з однаковою молодшою частиною адреси, що є зміщенням всередині сторінки. Один рядок у певний момент містить копію

тільки однієї з цих сторінок. Адресу (номер) рядка в кеш-пам'яті називають *індексом* (index).

Інформацію про те, яку саме сторінку основної пам'яті займає даний рядок, тобто старшу частину адреси чи номер сторінки, містить тег. Пам'ять тегів має кількість комірок, яка дорівнює кількості рядків кеша, а її розрядність повинна бути достатньою, щоб вмістити старші біти адреси кешованої пам'яті, які не потрапили на шину адреси кеш-пам'яті. Крім адресної частини тега кожний рядок кеша відображає біти ознак дійсності та модифікування даних.

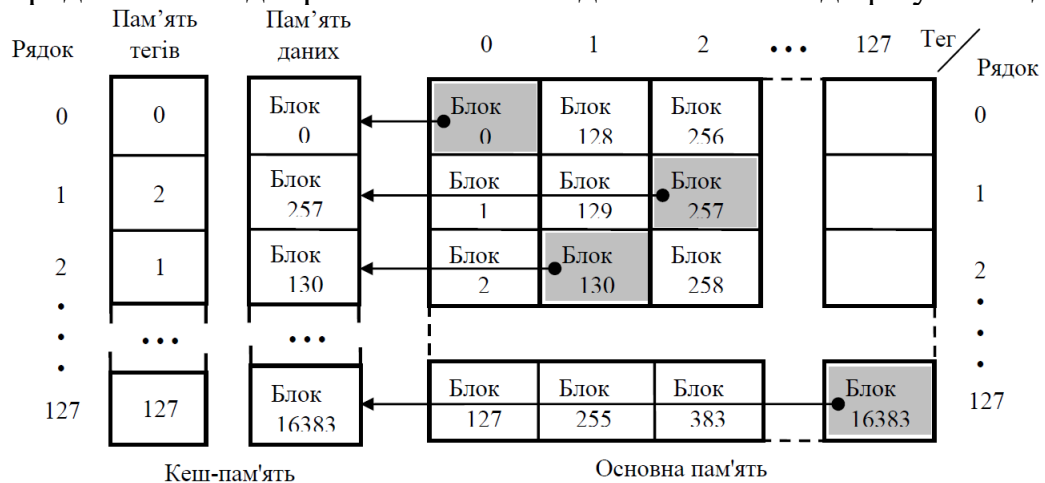


Рисунок 3.13 – Організація кеш-пам'яті з прямим відображенням

На цьому рисунку показана основна пам'ять з 14-розрядною адресою блоків, тобто адреса блоків може змінюватись від 0 до 16383. Логіка кеш-пам'яті інтерпретує ці 14 біт як 7-розрядний тег і 7-розрядне поле рядка.

На початку кожного звернення до кеш-пам'яті контролер спочатку зчитує комірку каталогу із даним індексом, порівнює біти тега зі старшими бітами адреси пам'яті та аналізує ознаку дійсності. Такий аналіз виконується в спеціальному *циклі стеження* (snooper cycle), який ще називають *циклом запиту* (inguire). Якщо в результаті аналізу з'ясується, що потрібний блок не знаходиться в кеші, то генерується або продовжується цикл звернення до основної пам'яті (випадок кеш-промаху). Коли ж є кеш-попадання, то запит обслуговується кеш-пам'яттю. У випадку кеш-промаху після зчитування основної пам'яті нові дані розміщуються в рядку кеша за умови, що він чистий, а в його тегу розташовуються старші біти адреси і встановлюється ознака дійсності даних. З основної пам'яті рядок переписується в кеш повністю, незалежно від обсягу даних, що вимагаються, оскільки ознака дійсності належить до всіх його байтів. Якщо контролер реалізує випереджаюче зчитування (read ahead), то в наступні вільні цикли шини поновиться і наступний рядок, якщо він був чистим. Читання "із запасом" дає змогу за необхідності здійснювати пакетний цикл зчитування з кеша через межу рядка.

Кеш розглянутого типу використовується у вторинному кеші більшості системних плат. Недоліком такої організації кеш-пам'яті є робота "вхолосту" (cache trashing), коли в процесі виконання програми процесора по черзі будуть потрібні блоки (сторінки) пам'яті, зміщені один стосовно іншого на величину,

кратну розміру сторінки. Чергове звернення замінюватиме дані, зчитані у попередньому зверненні, які будуть необхідні в наступному. Таким чином, це буде суцільна низка кеш-промахів. Зменшує кількість кеш-попадань також переключення сторінок у багатозадачних обчислювальних системах. Оскільки різні задачі претендуватимуть на одні й ті самі рядки кеша, то збільшення його розмірів за архітектури прямого відображення не дає суттєвого підвищення ефективності. Підвищити її, не збільшуючи ємність кеша, можна тільки зміною структури кеш-пам'яті.

Повністю асоціативний кеш. У повністю асоціативному кеші, на відміну від попередньої архітектури, будь-який його рядок може відображати будь-який блок пам'яті. Це істотно підвищує ефективність використання його обмеженої ємності.

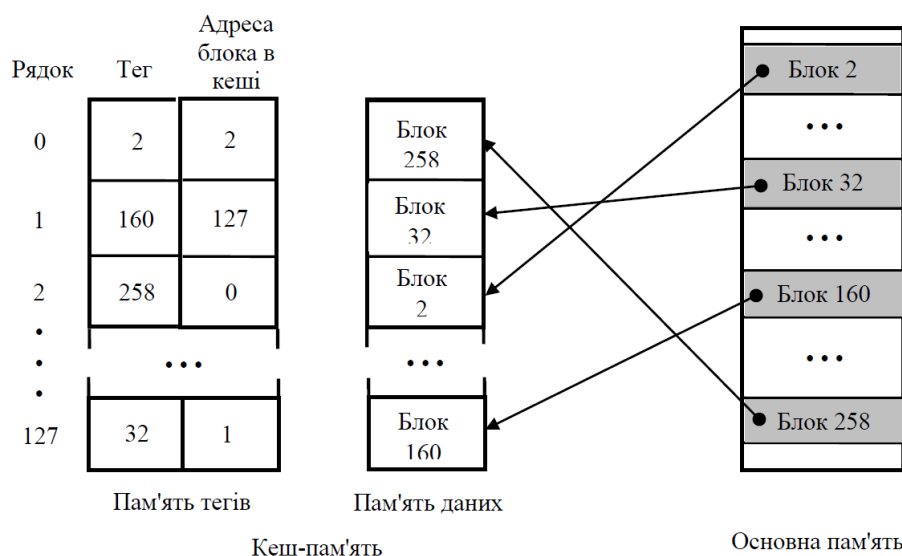


Рисунок 3.14 – Кеш-пам'ять з асоціативним відображенням.

Такий кеш містить два банки: пам'ять даних і пам'ять тегів. Всі біти адреси кешованого блока, за винятком бітів, які визначають розташування даних у рядку (зміщення), зберігаються в пам'яті тегів. Повна адреса ОП для повністю асоціативної пам'яті ділиться на дві частини. Старші розряди є тегом і визначають номер блока даних в ОП, а молодші – зміщення в рядку кеша.

Довжина блока ОП, як і раніше, дорівнює довжині рядка кеш-пам'яті. Як пам'ять тегів у даній архітектурі використовується асоціативна пам'ять. При цьому кількість комірок асоціативної пам'яті тегів дорівнює кількості рядків кеша, тобто кожному рядку кеша відповідає однойменна комірка тега.

Під час запису блока даних з ОП в рядок кеша одночасно в тег записується старша частина адреси (номер рядка). При цьому для визначення наявності даних кеш-пам'яті, які вимагаються, необхідне порівняння тегів усіх рядків зі старшою частиною адреси, а не одного або кількох, як у разі прямого відображення або набірно-асоціативній архітектурі. Таким чином, відпадає необхідність послідовного перебирання комірок пам'яті тегів. Виконується тільки паралельний аналіз усіх комірок.

Читання слова з кеш-пам'яті починається з асоціативного пошуку тега адресованого слова в асоціативній пам'яті тегів по коду, який є старшою частиною адреси ОП. Якщо під час асоціативного пошуку знаходиться тег адресованого слова, то воно зчитується з рядка кеша, номер якого визначається номером комірки тега, що збігся, а положення в рядку – зміщенням. Якщо результат асоціативного пошуку негативний, то адресованого слова немає в кеш-пам'яті і його необхідно зчитувати з ОП. При цьому разом з передачею адресованого слова в процесор в кеш-пам'ять переписується весь блок, в якому знаходиться це слово.

Недоліком цієї архітектури кеш-пам'яті є велика апаратна складність пошуку інформації, зв'язана з необхідністю виконувати порівняння усіх тегів паралельно по всіх комірках асоціативної пам'яті. З підвищенням ємності кеш-пам'яті, відповідно і асоціативної пам'яті тегів, зростає її складність і вартість. Тому така архітектура використовується тільки для побудови кеш-пам'яті невеликої ємності (первинного кеша в деяких процесорах).

Набірно-асоціативний кеш. Набірно-асоціативна архітектура кеша дає можливість кожній сторінці основної кешованої пам'яті претендувати на один з кількох рядків кеша, об'єднаних в набір (set). В кеші такої архітектури є кілька паралельно і погоджено працюючих каналів прямого відображення, де контролер кеша приймає рішення про те, в який з рядків набору помістити черговий блок даних. Розглянемо організацію кеш-пам'яті з множинно-асоціативним відображенням.

Множинно-асоціативне відображення відноситься до групи методів частково-асоціативного відображення. Воно є одним з можливих компромісів, що поєднує переваги прямого і асоціативного способів відображення і, певною мірою, вільним від їх недоліків. Кеш-пам'ять (як тегів, так і даних) розбивається на v підмножин (надалі називатимемо такі підмножини модулями), кожна з яких містить k рядків (прийнято говорити, що модуль має k входів). Залежність між модулем і блоками ОП така ж, як і при прямому відображенні: на рядки, що входять в модуль i , можуть бути відображені тільки цілком певні блоки основної пам'яті, відповідно до співвідношення $i = j \bmod v$, де j - адреса блока ОП. У той же час розміщення блоків по рядках модуля – довільне, і для пошуку потрібного рядка в межах модуля використовується асоціативний принцип.

На рис. 4 показаний приклад чотириходової кеш-пам'яті з множинно-асоціативним відображенням. Пам'ять даних кеш-пам'яті розбита на 32 модулі по 4 рядки в кожному. Пам'ять тегів містить 32 комірки, в кожній з яких може зберігатися 4 значення тегів (поодинокі на кожен рядок модуля). 14-розрядна адреса блока ОП подається у вигляді двох полів: 9-розрядного поля тега і 5-розрядного поля номера модуля.

Номер модуля однозначно указує на один з модулів кеш-пам'яті. Він також дозволяє визначити номери тих блоків ОП, які можна відображати на цей модуль. Такими є блоки ОП, номери яких під час ділення на 25 дають у залишку число, що збігається з номером даного модуля кеш-пам'яті. Так,

блоки 0, 32, 64, 96 і так далі в основній пам'яті відображаються на модуль з номером 0; блоки 1, 33, 65, 97 і так далі відображаються на модуль 1 і т. д. Будь-який з блоків у послідовності може бути завантажений у будь-який з чотирьох рядків відповідного модуля.

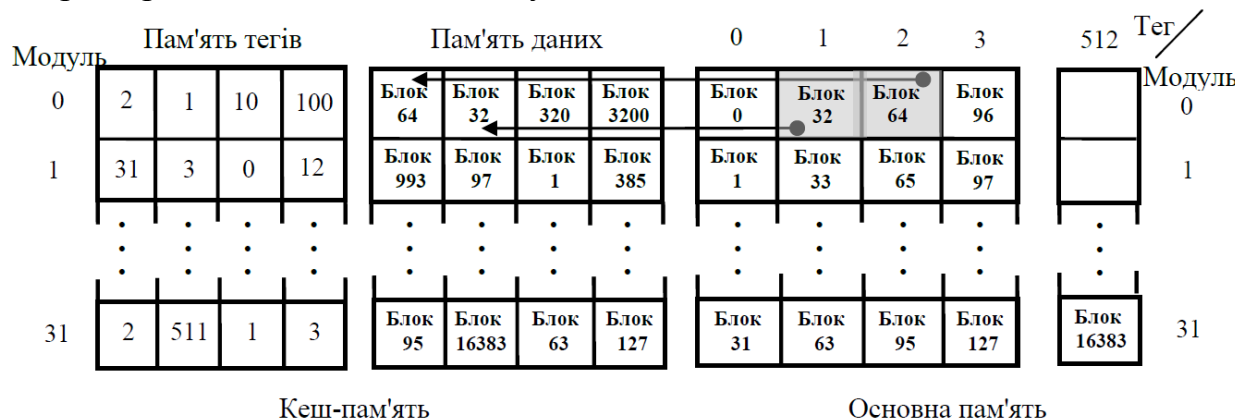


Рисунок 3.15 – Кеш-пам'ять з множинно-асоціативним відображенням.

У випадку такої постановки роль тега виконують 9 старших розрядів адреси блока ОП, в яких міститься порядковий номер блока в послідовності блоків, що відображаються на один і той же модуль кеш-пам'яті. Наприклад, блок 65 у послідовності блоків, що відображаються на модуль 1, має порядковий номер 2 (відлік ведеться від 0).

Під час звернення до кеш-пам'яті 5-розрядний номер модуля указує на конкретну комірку пам'яті тегів (це відповідає прямому відображенню). Далі проводиться паралельне порівняння кожного з чотирьох тегів, що зберігаються в цій комірці, з полем тега адреси, що поступила, тобто пошук потрібного тега серед чотирьох можливих здійснюється асоціативно.

У граничних випадках, коли $v=m$, $k=1$, множинно-асоціативне відображення зводиться до прямого, а при $v=1$, $k=m$ – до асоціативного.

Слід зазначити, що саме цей спосіб відображення найширше розповсюджений у сучасних мікропроцесорах.

Лекція 3 (2 год.) Організація віртуальної пам'яті в комп'ютерах на базі процесорів з архітектурою x86.

План лекції:

1. Загальні принципи організації віртуальної пам'яті.
2. Сторінкова організація віртуальної пам'яті.
3. Сегментно-сторінкова організація віртуальної пам'яті.

Для більшості типових застосувань обчислювальних машин характерна ситуація, коли розміщення всієї програми в ОП неможливо через її великий розмір. У цьому, проте, і немає принципової необхідності, оскільки в кожен момент часу «увага» машини концентрується на визначених порівняно невеликих ділянках програми. Таким чином, в ОП досить зберігати тільки використовувані в даний період частини програм, а решта частин може

розташовуватися на зовнішніх ЗП (ЗЗП). Складність подібного підходу в тому, що процеси звернення до ОП і ЗЗП істотно розрізняються, і це ускладнює завдання програміста. Виходом з такої ситуації була поява в 1959 році ідеї *віртуалізації пам'яті*, під якою розуміється метод автоматичного управління ієрархічною пам'яттю, при якому програмістові здається, що він має справу з єдиною пам'яттю великої ємності і високої швидкодії. Цю пам'ять називають *віртуальною* (що здається) *пам'яттю*. За своєю суттю віртуалізація пам'яті є способом апаратної і програмної реалізації концепції ієрархічної пам'яті.

В рамках ідеї віртуалізації пам'яті ОП розглядається як лінійний простір N адрес, названий *фізичним простором* пам'яті. Для завдань, де потрібно більш ніж N комірок, надається значно більший простір адрес (зазвичай рівний загальній ємності всіх видів пам'яті), названий *віртуальним простором*, у загальному випадку не обов'язково лінійний. Адреси віртуального простору називають *віртуальними*, а адреси фізичного простору – *фізичними*.

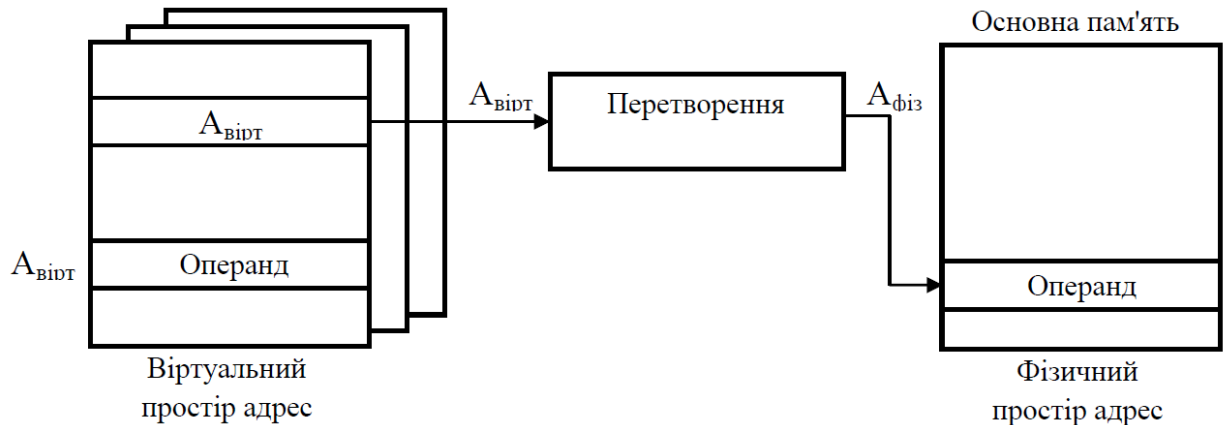


Рисунок 3.16 – Відображення віртуальної адреси на фізичну.

Програма пишеться у віртуальних адресах, але оскільки для її виконання потрібно, щоб оброблювані команди і дані знаходилися в ОП, потрібно, щоб кожній віртуальній адресі відповідала фізична. Таким чином, у процесі обчислень необхідно, перш за все, переписати з ЗЗП в ОП ту частину інформації, на яку вказує віртуальна адреса (відобразити віртуальний простір на фізичний), після чого перетворити віртуальну адресу у фізичну.

Серед систем віртуальної пам'яті можна виділити два класи: системи з фіксованим розміром блоків (сторінкова організація) і системи із змінним розміром блоків (сегментна організація). Обидва варіанти зазвичай суміщають (сегментно-сторінкова організація).

Цілям перетворення віртуальних адрес у фізичні служить сторінкова організація пам'яті. Її ідея полягає в розбитті програми на частини рівної величини, що називаються *сторінками*. Розмір сторінки зазвичай вибирають в межах 4-8 Кбайт, але так, щоб він був кратний ємності одного сектора магнітного диска. Віртуальний і фізичний адресні простори розбиваються на блоки розміром у сторінку. Блок основної пам'яті, відповідний сторінці, часто називають *сторінковим кадром* або *фреймом* (page frame). Сторінкам

віртуальної і фізичної пам'яті привласнюють номери. Процес доступу до даних за їх віртуальною адресою ілюструє рис. 6.

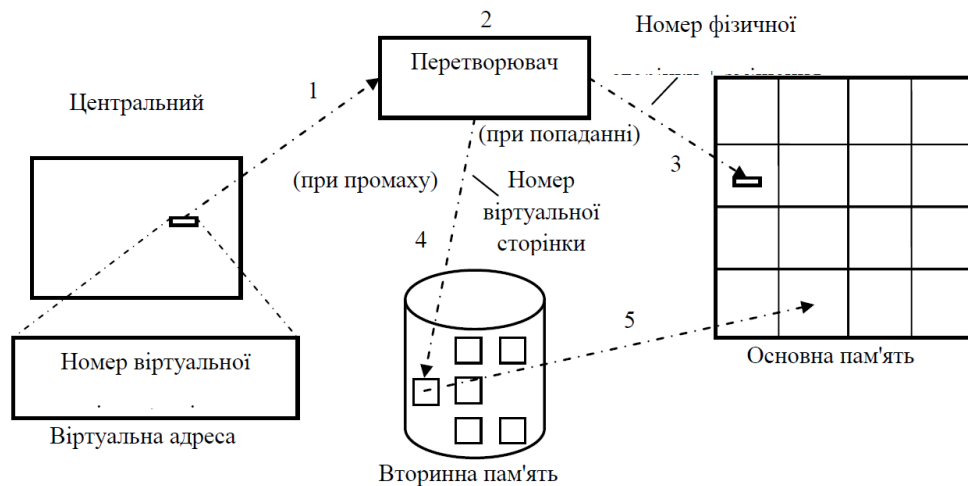


Рисунок 3.17 – Сторінкова організація віртуальної пам'яті.

Центральний процесор звертається до комірки, вказавши її віртуальну адресу (1), яка складається з номера віртуальної сторінки і зсуву щодо її початку. Ця адреса поступає в систему перетворення адрес (2), з метою отримання з неї фізичної адреси комірки в основній пам'яті (3). Оскільки зсув у віртуальній і фізичній адресі однаковий, перетворенню піддається лише номер сторінки. Якщо перетворювач виявляє, що потрібна фізична сторінка відсутня в основній пам'яті (відбувся промах або сторінковий збій), то потрібна сторінка зчитується із зовнішньої пам'яті і заноситься в ОП (4, 5).

Перетворювач адрес – це частина операційної системи, що транлює номер віртуальної сторінки в номер фізичної сторінки, розташованої в основній пам'яті, а також апаратура, що забезпечує цей процес і що дозволяє прискорити його. Перетворення здійснюється за допомогою так званої *сторінкової таблиці*. За відсутності потрібної сторінки в ОП перетворювач адрес виробляє ознаку сторінкового збою, по якому операційна система припиняє обчислення, поки потрібна сторінка не буде зчитана з вторинної пам'яті і поміщена в основну.

Віртуальний простір повністю описується двома таблицями: сторінковою таблицею і картою диска (вважатимемо, що вторинна пам'ять реалізована на магнітних дисках). Таблиця сторінок визначає, які віртуальні сторінки знаходяться в основній пам'яті і в яких фізичних фреймах, а карта диска містить інформацію про сектори диска, де зберігаються віртуальні сторінки на диску.

Число записів у сторінковій таблиці (СТ) в загальному випадку рівне кількості віртуальних сторінок. Кожен запис містить поле номера фізичної сторінки (НФС) і чотири ознаки: **V**, **R**, **M** і **A**.

Ознака присутності V встановлюється в одиницю, якщо віртуальна сторінка в даний момент знаходиться в основній пам'яті. В цьому випадку в полі номера фізичної сторінки знаходиться відповідний номер. Якщо **V** = 0, то у разі спроби звернутися до даної віртуальної сторінки перетворювач адреси

генерує сигнал сторінкового збою (page fault), і операційна система робить дії із завантаження сторінки з диска в ОП, звертаючись для цього до карти диска.

Ознака використання сторінки R встановлюється у разі звернення до даної сторінки. Ця інформація використовується в алгоритмі заміщення сторінок для вибору тієї з них, яку можна найбільш безболісно видалити з ОП, щоб звільнити місце для нової. Проблеми заміщення інформації в ОП вирішуються так само, як і для кеш-пам'яті.

Оскільки в ОП знаходяться лише копії сторінок, а їх оригінали зберігаються на диску, необхідно забезпечити ідентичність оригіналів і копій. У ході обчислень вміст окремих сторінок може змінюватися, що фіксується шляхом установки в одиницю *ознаки модифікації M*. Під час видалення сторінки з ОП перевіряється стан ознаки *M*. Якщо $M = 1$, то перед видаленням сторінки з основної пам'яті її необхідно переписати на диск, а при $M = 0$ цього можна не робити.

Ознака прав доступу A служить цілям захисту інформації і визначає, який вид доступу до сторінки дозволений: тільки для читання, тільки для запису, для читання і для запису.

У разі сторінкової організації передбачається, що віртуальна пам'ять – це безперервний масив з наскрізною нумерацією слів, що не завжди можна визнати оптимальним. Зазвичай програма складається з декількох частин – кодової, інформаційної та стекової. Оскільки заздалегідь невідомі довжини цих складових, то зручно, щоб під час програмування кожна з них мала власну нумерацію слів, відлічуваних з нуля. Для цього організовують систему *сегментованої пам'яті*, виділяючи у віртуальному просторі незалежні лінійні простори змінної довжини, що називаються *сегментами*. Кожен сегмент є окремою логічною одиницею інформації, що містить сукупність даних або програмний код і розташована в адресному просторі користувача. В кожному сегменті встановлюється своя власна нумерація слів, починаючи з нуля. Віртуальна пам'ять також розбивається на сегменти, з незалежною адресацією слів усередині сегмента. Кожній складовій програми виділяється сегмент пам'яті. Віртуальна адреса визначається номером сегмента і адресою всередині сегмента. Для перетворення віртуальної адреси у фізичну використовується спеціальна *сегментна таблиця*.

Недоліком такого підходу є те, що неоднаковий розмір сегментів приводить до неефективного використання ОП. Так, якщо ОП заповнена, то у разі заміщення одного з сегментів потрібно витіснити такий, розмір якого дорівнює або більше розміру нового. У разі багатократного повтору подібних дій в ОП залишається багато вільних ділянок, недостатніх за розміром для завантаження повного сегмента. Вирішенням проблеми служить *сегментно-сторінкова організація пам'яті*. В ній розмір сегмента вибирається не довільно, а задається кратним розміру сторінки.

Структуру віртуальної адреси і процес перетворення її у фізичну адресу ілюструє рис. 7.

Сегмент може містити те або інше, але обов'язково ціле число сторінок, навіть якщо одна із сторінок заповнена частково. Виникає певна ієрархія в організації доступу до даних, що складається з трьох ступенів: сегмент > сторінка > слово. Цій структурі відповідає ієрархія таблиць, які служать для перекладу віртуальних адрес у фізичні. В сегментній таблиці програми перераховуються всі сегменти даної програми з вказівкою початкових адрес СТ, які відносяться до кожного сегмента. Кількість сторінкових таблиць дорівнює числу сегментів і будь-яка з них визначає розташування кожної із сторінок сегмента в пам'яті, які можуть розташовуватися не підряд – частина сторінок може знаходитися в ОП, останні – у зовнішній пам'яті.

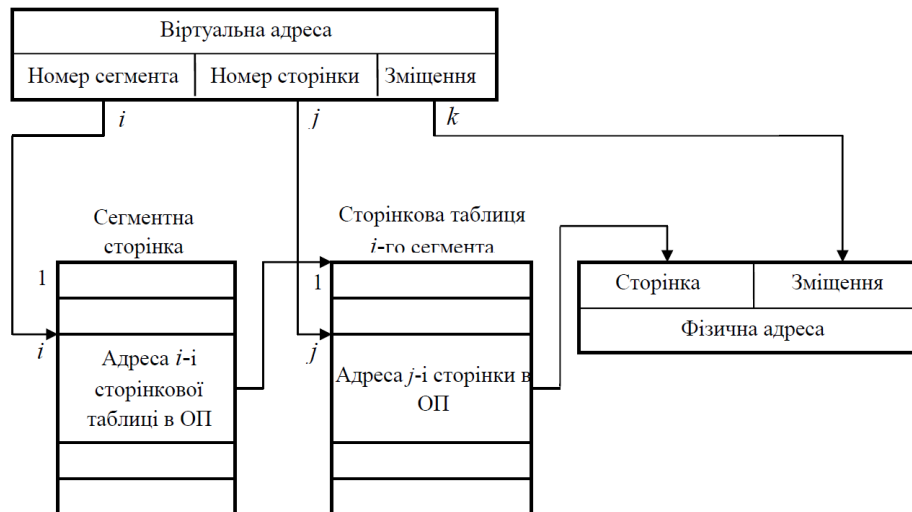


Рисунок 3.18 – Перетворення адреси при сегментно-сторінковій організації пам'яті.

Для отримання фізичної адреси необхідний доступ до сегментної та однієї із сторінкових таблиць, тому перетворення адреси може займати багато часу.

Контрольні запитання:

1. Яка інформація зберігається у кеш-пам'яті?
2. Які завдання має вирішувати система кешування?
3. Дайте визначення коефіцієнта потрапляння до кешу.
4. Який кеш називається повністю асоціативним?
5. Який кеш називається множино-асоціативним?
6. Якими є основні цілі створення віртуального адресного простору?

Тема 4.1 Рівні паралелізму. Класифікація архітектури комп'ютерних систем.

Лекція 1 (2 год.) Рівні паралелізму комп'ютерних систем.

План лекції:

1. Способи організації паралельної обробки.
2. Рівні паралелізму.
3. Особливості реалізації паралелізму рівня завдань та паралелізму рівня програм.
4. Особливості реалізації паралелізму рівня команд.

В умовах постійно зростаючих вимог до продуктивності обчислювальної техніки все більш явними стають обмеження класичної фон-нейманівської архітектури. Подальший розвиток обчислювальної техніки зв'язаний з переходом до паралельних обчислень як у рамках однієї обчислювальної машини, так і шляхом створення багатопроцесорних систем і мереж, які об'єднують велику кількість окремих процесорів або окремих обчислювальних машин. Для такого підходу замість терміну «обчислювальна машина» більш підходить термін «обчислювальна система» (ОС). Головною особливістю такої системи є наявність у ній засобів, які реалізують паралельну обробку інформації.

Паралельна обробка інформації являє собою одночасне рішення двох або більшої кількості частин однієї й тієї ж програми двома чи більшою кількістю ЕОМ (процесорними елементами) обчислювальної системи. Реалізують три основних способи організації паралельної обробки:

- 1) суміщення у часі різноманітних етапів різних задач – це мульти-програмна обробка, яка широко використовується як у однопроцесорних ЕОМ, так і в складних ОС;
- 2) одночасне розв'язання різноманітних задач або частин однієї задачі (можливо тільки за наявності декількох обробляючих пристроїв);
- 3) конвеєрна обробка інформації.

Перші два способи використовують особливості паралельних задач або потоків задач, що дозволяє здійснювати той або інший паралелізм. Перший тип паралелізму – це *природний паралелізм незалежних задач*, який полягає у тому, що в систему поступає безперервний потік не зв'язаних між собою задач. У цьому випадку розв'язання будь-якої задачі не залежить від результатів розв'язання інших задач, що дозволяє підвищити продуктивність ОС у разі використання декількох обробляючих пристроїв.

Одним з найбільш розповсюджених типів паралелізму є *паралелізм незалежних гілок*. Суть його полягає у виділенні окремих незалежних частин великої задачі (гілок програми), які можуть виконуватись паралельно окремими обробляючими пристроями незалежно один від одного. Причому обробляючі пристрої ОС функціонують в однопрограмному режимі

паралельної обробки інформації. Двома незалежними гілками програми визнаються гілки програми, що відповідають таким умовам:

- ні одна з вхідних для гілки програми величин не є вихідною величиною іншої програми (відсутність функціональних зв'язків);
- для двох гілок програми не повинен здійснюватися запис у одні й ті ж комірки пам'яті (відсутність зв'язку по оперативній пам'яті);
- умови виконання однієї гілки не залежать від результатів, що отримані під час виконання іншої гілки (незалежність по управлінню);
- обидві гілки повинні виконуватись у різних блоках програми (програмна незалежність).

Виділення незалежних гілок широко використовується під час паралельної обробки в задачах матричної алгебри, лінійного програмування, спектральної обробки сигналів, прямого та зворотного перетворення Фур'є та ін.

Методи та засоби реалізації паралелізму залежать від того, на якому рівні він повинен забезпечуватись. Зазвичай розрізняють такі *рівні паралелізму*:

- ***рівень завдань*** – декілька незалежних завдань одночасно виконуються на різних процесорах, які практично не взаємодіють один з одним. Цей рівень реалізується в ОС з множиною процесорів у багатозадачному режимі.
- ***рівень програм*** – частини однієї задачі виконуються множиною процесорів. Даний рівень досягається на паралельних ОС.
- ***рівень команд*** – виконання команди розділяється на фази, а фази декількох послідовних команд можуть бути перекриті за рахунок конвеєризації. Даний рівень використовується в ОС з одним процесором.
- ***рівень бітів (арифметичний рівень)*** – біти слова обробляються один за одним. Цей процес має назву *біт-послідовна операція*. Якщо біти слова обробляються одночасно, кажуть про *біт-паралельну операцію*. Даний рівень реалізується в звичайних і суперскалярних процесорах.

Паралелізм рівня завдання можливий між незалежними завданнями або їх фазами. Основним засобом реалізації паралелізму на рівні завдань є багатопроцесорні і багатомашинні обчислювальні системи, в яких завдання розподіляються за окремими процесорами або машинами. Однак, якщо кожне завдання трактувати як сукупність незалежних задач, реалізація даного рівня можлива і в рамках однопроцесорної ОС. У цьому випадку декілька завдань можуть одночасно знаходитись в основній пам'яті ОС, за умови, що в кожному момент виконується тільки одне з них. Коли завдання, яке виконується, потребує вводу/виводу, ця операція запускається, а до її завершення інші ресурси ОС передаються другому завданню. По завершенні вводу/виводу ресурси ОС повертаються до завдання, яке ініціювало цю операцію. В цьому випадку паралелізм забезпечується за рахунок того, що центральний процесор

і система вводу/виводу функціонують одночасно і обслуговують різні завдання.

Паралелізм виникає також, коли у незалежних завдань, які виконуються в ОС, є декілька фаз, наприклад обчислення, запис у графічний буфер, системні виклики. За те, як різні завдання впорядковуються і витрачають загальні ресурси відповідає операційна система.

Паралелізм рівня програм. Про паралелізм на рівні програми можна говорити у двох випадках. По перше, коли в програмі можна виділити незалежні ділянки, які допустимо виконувати паралельно. Другий тип паралелізму програм можливий у межах окремого програмного циклу, якщо в ньому окремі ітерації не залежать один від одного. Програмний паралелізм можна реалізувати за рахунок великої кількості процесорів або множини функціональних блоків.

Загальна форма паралелізму на рівні програм організується розбиттям даних, які програмуються на підмножини. Цей розподіл називають *декомпозицією області* (domain decomposition), а паралелізм, який при цьому виникає, має назву *паралелізму даних*. Підмножини даних призначаються різним обчислювальним процесам, і цей процес має назву *розподілення даних* (data distribution). Процесори виділяються певним процесам або за ініціативою програми, або в процесі роботи операційною системою. На кожному процесорі може виконуватись більше одного процесу.

Паралелізм рівня команд. Паралелізм на рівні команд має місце, коли обробка декількох команд або виконання різних етапів однієї і тієї ж команди може перекриватись у часі. Розробники обчислювальної техніки ще здавна зверталися до методів, відомих під загальною назвою «поєднання операцій», при якому апаратура ОМ у будь-який момент часу виконує одночасно більше однієї операції. Цей загальний принцип включає два поняття: *паралелізм* і *конвеєризацію*. Хоча у них багато загального і їх часто важко розрізнити на практиці, ці терміни відображають два принципово різних підходи.

У першому варіанті поєднання операцій досягається за рахунок того, що в складі обчислювальної системи окремі пристрої присутні в декількох копіях. Так, до складу процесора може входити декілька АЛП, і висока продуктивність забезпечується за рахунок одночасної роботи всіх цих АЛП.

Під час організації паралельного обчислювального процесу виникає задача вибору побудови розкладу.

Розклад паралельних обчислювальних процесів визначає порядок виконання програми в ОС, включаючи розподіл частин програми по обробляючих пристроях (процесорах, ОМ), і служить основою алгоритмів планувальника операційної системи та різноманітних управляючих програм. Як критерії оптимальних розкладів для паралельної програми можна назвати:

- мінімізацію часу виконання програми;
- мінімізацію кількості потрібних пристроїв обробки;
- мінімізацію середнього часу закінчення виконання завдань;
- максимізацію завантаження пристроїв ОС;

- мінімізацію часу простоювання пристроїв.
Найчастіше використовують перший критерій.

Лекція 2 (2 год.) Класифікація архітектури комп'ютерних систем.

План лекції:

1. Класифікація архітектури комп'ютерних систем за Флінном.
2. Організація комп'ютерної архітектури відповідно до класифікації Фліна.
3. Інші класифікації архітектури комп'ютерних систем.

Існує декілька класифікацій архітектур комп'ютерних систем. Найбільш вдалою є *класифікація М. Флінна*.

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. 4.1, рис. 4.1).

Класифікація здійснена з погляду не структури, а того, як у комп'ютері його машинні команди взаємодіють з даними.

Таблиця 4.1 – Класифікація архітектури комп'ютерних систем

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)

SISD (Single Instruction Stream/Single Data Stream) - одиничний потік команд і одиничний потік даних. Представник цього класу – класичний фон-нейманівський комп'ютер. Команди обробляються послідовно і кожна команда ініціює одну операцію з одним потоком даних. До цього класу комп'ютерів можна віднести також конвеєрні комп'ютери. Деякі спеціалісти вважають, що до SISD-систем можна віднести і векторно-конвеєрні ОС, якщо розглядати вектор як неподільний елемент даних для відповідної команди.

MISD (Multiple Instruction Stream/Single Data Stream) – множинний потік команд і одиничний потік даних. В архітектурі присутня множина процесорів, які обробляють один і той же потік даних. Ряд дослідників відносять до даного класу конвеєрні системи. Прийнято вважати, що доки даний клас незадіяний, однак може бути корисним для розробки принципово нових концепцій побудови обчислювальних систем.

SIMD (Single Instruction Stream/Multiple Data Stream) – одиничний потік команд і множинний потік даних. Ця архітектура дозволяє виконати одну

арифметичну операцію відразу над багатьма даними – елементами вектору. Представниками цього класу є системи з матрицею процесорів, де один управляючий пристрій контролює множину процесорних елементів. Усі процесорні елементи отримують від пристрою управління однакову команду і виконують її над власними локальними даними. В цей клас можуть бути включеними і векторно-конвеєрні ОС, якщо кожний елемент вектора розглядати як окремий елемент даних.

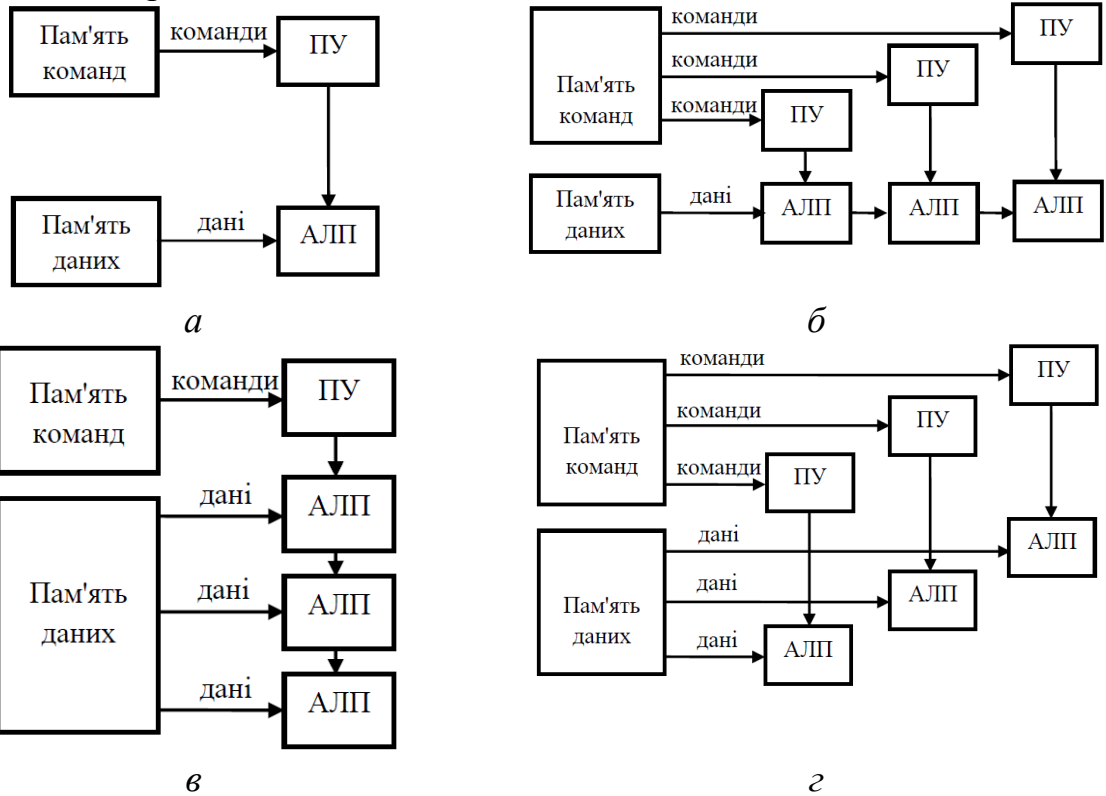


Рисунок 4.1 – Архітектура комп'ютерних систем за Флінном: *а* – SISD; *б* – MISD; *в* – SIMD; *г* – MIMD

MIMD (Multiple Instruction Stream/Multiple Data Stream) – множинний потік команд і множинний потік даних. До цього класу відносяться системи з множиною пристроїв обробки команд, які об'єднані в єдиний комплекс і кожний працює з власним потоком команд. Цей клас надзвичайно широкий, бо включає в себе різного роду мультипроцесорні системи.

Класифікація Флінна надто загальна, вона має деякі недоліки, наприклад відносить усі паралельні комп'ютери, крім мультипроцесорних, до одного класу і не вказує ніякої відмінності між конвеєрними комп'ютерами і матрицею МП.

Використовуються й інші класифікації архітектури, зокрема систематика Ф. Шара, структурна систематика Р. Хокні та К. Джессхоупа. Відповідно до неї на першому рівні всі обчислювальні системи поділяються за принципом множинності (кількості) на однокomp'ютерні та багатокомп'ютерні. Обчислювальні системи з одним комп'ютером, у свою чергу, поділяються на OM з одним конвеєрним МП та з багатьма МП.

Перші з них є традиційними послідовними комп'ютерами, а другі утворюють клас паралельних комп'ютерів, які поділяються на конвеєрні, неконвеєрні та мікропроцесорні матриці.

Прикладом однієї з перших неконвеєрних обчислювальних машин з паралелізмом є комп'ютер CDC – 6600, побудований на основі декількох скалярних процесорів.

Конвеєрні ЕОМ поділяються на такі, що виконують тільки скалярні команди, наприклад комп'ютери CDC – 7800, FPS AP-120B, і такі, що виконують векторні команди. Комп'ютери, що використовують векторні команди, поділяють, у свою чергу, на комп'ютери із спеціалізованим конвеєром, наприклад CRAY-1, та з універсальним конвеєром – комп'ютер CYBER 205.

Комп'ютери з класу машин з матрицею процесорів поділяють за зв'язаністю процесорів у матриці, розрядністю тощо. Першими машинами такого типу були ILLIAC-IV, BSP, STA-RAN, ICL DAP, OMEN та ін.

Контрольні запитання:

1. Які існують способи організації паралельної обробки?
2. Які рівні паралелізму існують?
3. У чому полягає паралелізм рівня програм?
4. У чому полягає паралелізм рівня завдань?
5. Як класифікуються комп'ютерні системи за Флінном?
6. Які ще існують класифікації комп'ютерних систем?

Тема 4.2 Обчислювальні системи класу SIMD та MIMD. Багатомашинні та багатопроцесорні комп'ютерні системи.

Лекція 1 (2 год.) Обчислювальні системи класу SIMD та MIMD.

План лекції:

1. Векторні і векторно–конвеєрні комп'ютерні системи.
2. Матричні комп'ютерні системи.
3. SMP-системи.
4. MPP-системи.

SIMD-системи були першими обчислювальними системами, що складаються з великого числа процесорів, і серед перших систем, де була досягнута продуктивність порядку GFLOPS. Згідно з класифікацією Флінна, до класу SIMD відносяться ОС, де множина елементів даних піддається паралельній, але однотипній обробці.

SIMD-системи багато в чому схожі на класичні фон-нейманівські ОМ: у них також є один пристрій управління, який забезпечує послідовне виконання команд програми. Відмінність стосується стадії виконання, коли загальна команда транслюється великій кількості процесорів (у простому випадку – АЛП), кожен з яких обробляє свої дані.

Раніше вже наголошувалася нечіткість класифікації Флінна, через що різні типи ОС можуть бути віднесені до того або іншого класу. Проте в теперішній час прийнято вважати, що клас SIMD складають векторні (векторно-конвеєрні), матричні, асоціативні, систолічні і VLIW-обчислювальні системи.

Векторні і векторно–конвеєрні комп'ютерні системи. Під час розв'язання на комп'ютері науково-технічних та інших задач часто зустрічається необхідність визначення значень однієї і тієї ж функції для групи даних, розташованих у комірках пам'яті (регістрах) з упорядкованими адресами (номерами). Такі набори даних називаються векторами.

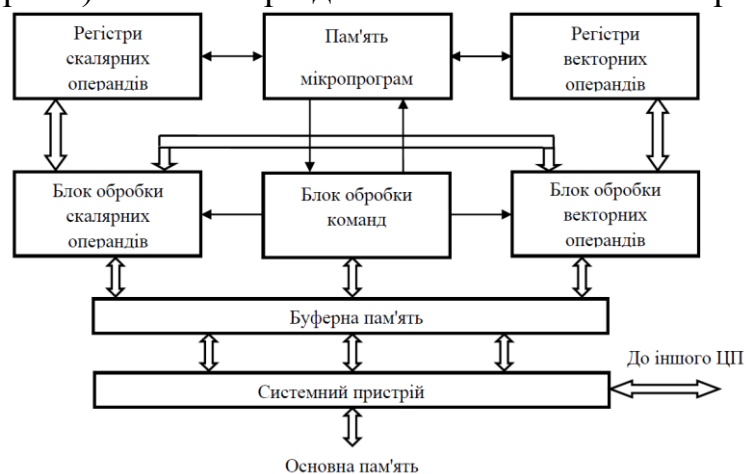


Рисунок 4.2 – Структурна схема векторного процесора.

З метою підвищення продуктивності комп'ютера до складу його процесора разом зі скалярними засобами включають засоби *векторної*

обробки даних або вводять спеціалізований векторний співпроцесор. Векторні засоби включають векторні команди, векторні регістри та іншу апаратуру для реалізації цих команд.

Засоби векторної обробки дозволяють за допомогою єдиної команди виконувати дії над усіма елементами масивів. Кожний елемент такого масиву (вектор) обробляється незалежно від інших елементів на конвеєрному операційному пристрої (ОП), який за рахунок спеціальної структурної організації може приймати в кожному такті пару елементів векторів-операндів і через деякий час, який називається *стартом конвеєра*, видавати відповідний елемент вектора-результату. Оскільки пари елементів векторів-операндів надходять у конвеєрний ОП в кожному такті, то через час, який визначає довжину часу старту конвеєра після запуску векторної операції, з виходу ОП кожного такту будуть видаватись елементи вектора-результату.

До апаратних векторних засобів відносяться:

- арифметичні конвеєри (кожний такий конвеєр може містити декілька незалежних спеціалізованих конвеєрних арифметичних пристроїв);
- векторні регістри, які зберігають векторну інформацію, що використовується для обробки в арифметичних конвеєрах;
- конвеєри завантаження-запису, які виконують обмін інформацією між векторними регістрами і оперативною пам'яттю ЕОМ.

Векторні засоби обробки інформації дозволяють збільшити продуктивність ЕОМ на кілька порядків. Для збільшення продуктивності ЕОМ необхідно дотримуватись таких вимог:

- використовувати арифметичні конвеєри з мінімально можливою довжиною такту;
- збільшувати число арифметичних конвеєрів;
- збільшувати число векторних команд, що паралельно опрацьовуються в одному арифметичному конвеєрі.

Векторні команди дозволяють однією командою дати розпорядження на виконання деякої операції (векторної операції) над елементами вектора, наприклад, поелементне додавання векторів та організовують і управління обчислювальним циклом.

У машині, яка має векторні команди, виконуються також звичайні команди над одиничними даними (скалярні команди).

Наявність векторних команд сприяє підвищенню швидкодії процесора за рахунок зменшення витрат часу на організацію обчислювального циклу.

Прикладами конвеєрно-векторних процесорів можуть служити спеціалізовані на виконання векторних і матричних операцій співпроцесори, при цьому продуктивність машини під час виконання векторних і матричних операцій збільшується до 30 Мфлоп/с.

Призначення **матричних комп'ютерних систем** багато в чому схоже з призначенням векторних комп'ютерних систем – обробка великих масивів даних. В основі матричних систем лежить *матричний процесор*, який складається з регулярного масиву процесорних елементів (ПЕ). Організація

системи такого типу, на перший погляд, достатньо проста. Вона має загальний управляючий пристрій, який генерує потік команд, та велику кількість ПЕ, які функціонують паралельно і обробляють кожний свій потік даних. Проте на практиці, щоб забезпечити достатню ефективність системи у ході розв'язання широкого кола задач, необхідно організовувати зв'язки між процесорними елементами так, щоб найбільш повно завантажити процесори роботою. Саме характер зв'язків між ПЕ і визначає різні властивості системи.

Матричний процесор інтегрує множину ідентичних функціональних блоків (ФБ), які логічно об'єднуються в матрицю і працюють у SIMD-стилі. Матриця процесорних елементів може бути конструктивно реалізована на одному кристалі або на декількох. Важливим є принцип функціонування – ФБ логічно скомпоновані у матрицю і функціонують синхронно, тобто існує тільки один потік команд для усіх блоків.



Рисунок 4.3 – Узагальнена модель матричної SIMD-системи.

Власне паралельна обробка множинних елементів даних виконується *масивом процесорів* (МПр). Єдиний потік команд, який управляє обробкою даних у масиві процесорів, генерується *контролером масиву процесорів* (КМПр). КМПр виконує послідовний програмний код, реалізує операції умовного і безумовного переходів, транслює в МПр команди, дані та сигнали управління. Команди обробляються процесорами в режимі жорсткої синхронізації. Сигнали управління використовуються для синхронізації команд і пересилок, а також для управління процесом обчислень, зокрема визначають, які процесори масиву повинні виконувати операцію, а які – ні. Команди, дані і сигнали управління передаються із КМПр у масив процесорів по *шині широкомовної розсилки*. Оскільки виконання операцій умовного переходу залежить від результатів обчислень, результати обробки даних у масиві транслюються в КМПр по *шині результату*.

У матричних SIMD-системах розповсюдження отримали два основних типи архітектурної організації масиву процесорних елементів.

У першому варіанті, який відомий як архітектура типу «*процесорний елемент – процесорний елемент*», N процесорних елементів (ПЕ) зв'язані між собою мережею з'єднань. Кожний ПЕ – це процесор з локальною пам'яттю. Процесорні елементи виконують команди, які отримують від КМПр по шині широкомовної розсилки, та обробляють дані як ті, що зберігаються у їх

локальній пам'яті, так і ті, що потрапляють з КМПр. Обмін даними між процесорними елементами здійснюється по *мережі з'єднань*, в той час як *шина вводу/виводу* служить для обміну інформацією між ПЕ і пристроями вводу/виводу. Для трансляції результатів з окремих ПЕ в контролер масиву процесорів служить *шина результату*. В багатьох алгоритмах дії по пересиланню інформації в більшості локальні, тобто виконуються між найближчими сусідами, тому дана архітектура, де кожний ПЕ зв'язаний тільки з сусідніми, достатньо ефективна.

Другий вид архітектури – *«процесор – пам'ять»*. У такій архітектурі двонаправлена мережа з'єднань зв'язує N процесорів з M модулями пам'яті. КМП управляє процесорами через широкомовну шину. Обмін даними між процесорами здійснюється як через мережу, так і через модулі пам'яті. Пересилка даних між модулями пам'яті та пристроями вводу/виводу забезпечується шиною вводу/ виводу. Для передачі даних з певного модуля пам'яті в КМП служить шина результату.

MIMD-системи володіють великою гнучкістю, зокрема вони можуть робити і як високопродуктивні однокористувацькі обчислювальні системи, і як багатопрограмні ОС, які паралельно виконують множину задач. Крім того, архітектура MIMD дозволяє найбільш ефективно розпорядитися усіма перевагами сучасної мікропроцесорної технології.

У MIMD-системі кожний процесорний елемент (ПЕ) виконує власну програму достатньо незалежно від інших ПЕ. У той же час ПЕ повинні якимось взаємодіяти один з одним. Відмінність у способі такої взаємодії визначає умовне ділення MIMD-систем на ОС з загальною пам'яттю та системи з розподіленою пам'яттю. В *системах із загальною пам'яттю*, які характеризують як *сильно зв'язані*, є загальна пам'ять даних і команд, до якої мають доступ усі ПЕ за допомогою загальної шини або мережі з'єднань.

До цього типу, зокрема, відносяться *симетричні мультипроцесори* (SMP, Symmetric Multiprocessor) та *системи з неоднорідним доступом до пам'яті* (NUMA, Non-Uniform Memory Access).

У *системах з розподіленою пам'яттю* або *слабо зв'язаних* багато-процесорних системах уся пам'ять розподілена між процесорними елементами, і кожний блок пам'яті доступний тільки «власному» процесору. Мережа з'єднань зв'язує процесорні елементи один з одним. Представниками цієї групи є *системи з масовим паралелізмом* (MPP, Massively Parallel Processing) та *класстерні обчислювальні системи*.

Поняття *симетричні мультипроцесорні обчислювальні системи*, так звані *SMP-системи* (Symmetric Multiprocessor), відноситься як до архітектури обчислювальної системи, так і до поведінки операційної системи, яка відображає дану архітектурну організацію. SMP можна визначити як обчислювальну систему, що має такі характеристики:

- є два або більше процесорів порівнянної продуктивності;
- процесори сумісно використовують основну пам'ять і функціонують в єдиному віртуальному і фізичному адресному просторі;

- усі процесори зв'язані між собою за допомогою шини або за іншою схемою так, що час доступу до пам'яті будь-якого з них є однаковий;
- усі процесори розділяють доступ до пристроїв вводу/виводу або через одні й ті ж канали, або через різні канали, які забезпечують доступ до одного й того ж зовнішнього пристрою;
- усі процесори здатні виконувати однакові функції (цим пояснюється термін «симетричні»);
- будь-який з процесорів може обслуговувати зовнішні переривання;
- обчислювальна система управляється інтегрованою операційною системою, яка організовує і координує взаємодію між процесорами та програмами на рівні завдань, задач, файлів і елементів даних.

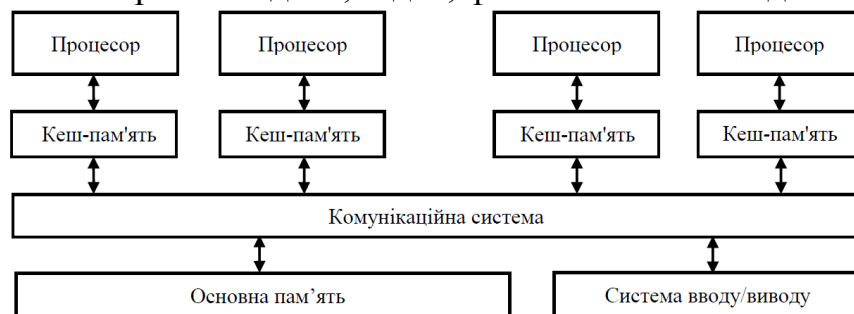


Рисунок 4.4 – Організація симетричної мультипроцесорної системи.

Типова SMP-система містить від двох до 32 ідентичних процесорів, у ролі яких звичайно використовують дешеві RISC-процесори. В останній час намітилася тенденція оснащення SMP-систем також і CISC-процесорами, зокрема Pentium.

Кожний процесор забезпечений локальною кеш-пам'яттю, яка складається з кеш-пам'яті першого (L1) та другого (L2) рівнів. Узгодженість вмісту кеш-пам'яті всіх процесорів забезпечується апаратними засобами. В деяких SMP-системах проблема когерентності знімається за рахунок загальної кеш-пам'яті. Застосування загальної кеш-пам'яті супроводжується збільшенням вартості і зменшенням швидкодії кеш-пам'яті.

Усі процесори обчислювальної системи мають рівноправний доступ до основної пам'яті і пристроїв вводу/виводу, що розділяються. Така можливість забезпечується комунікаційною системою. Звичайно процесори взаємодіють між собою через основну пам'ять. В деяких SMP-системах передбачається також прямий обмін сигналами між процесорами.

Основною ознакою, за якою обчислювальну систему відносять до *архітектури з масовою паралельною обробкою* (MPP - Massively Parallel Processing), служить кількість процесорів n . Строгої межі не існує, але при n 128 вважається, що це вже MPP, а при n 32 – ще ні.

Головні особливості, за якими обчислювальну систему відносять до класу MPP, можна сформулювати таким чином:

- стандартні мікропроцесори;
- фізично розподілена пам'ять;

- мережа з'єднань з високою пропускнуою здатністю і малими затримками;
- добра масштабованість (можливість варіювання числом процесорів (до тисяч процесорів));
- асинхронна MIMD-система з пересилкою повідомлень;
- програма являє собою множину процесів, які мають окремі адресні простори.

Характерна риса MPP-систем – наявність єдиного управляючого пристрою (процесора), що розподіляє завдання між множиною підлеглих пристроїв. Схема взаємодії в загальних рисах достатньо проста:

- центральний управляючий пристрій формує чергу завдань, кожному з яких призначається деякий рівень пріоритету;
- по мірі звільнення підлеглих пристроїв їм передаються завдання з черги;
- підлеглі пристрої оповіщають центральний процесор про хід виконання завдань, зокрема про завершення виконання або про потребу в додаткових ресурсах;
- у центрального пристрою є засоби для контролю роботи підлеглих процесорів, зокрема для виявлення нештатних ситуацій, переривання виконання завдань у випадку появи більш пріоритетної задачі та ін.

Завдяки властивості масштабованості, MPP-системи на сьогодні є лідерами по досягнутій продуктивності. З іншого боку, розпаралелювання в MPP-системах є складною задачею. Ефективність розпаралелювання у багатьох випадках сильно залежить від деталей архітектури MPP-системи. Для синхронізації паралельно виконуваних процесів необхідний обмін повідомленнями, які повинні доходити з будь-якого вузла системи у будь-який інший вузол. Час передачі інформації від вузла до вузла залежить від стартової затримки та швидкості передачі. Продуктивність процесорів набагато більше пропускнуої здатності каналів зв'язку, тому інфраструктура каналів зв'язку в MPP-системах є найбільш проблемною.

Лекція 2 (2 год.) Багатомашинні та багатопроцесорні комп'ютерні системи.

План лекції:

1. Принципи організації багатомашинної обчислювальної системи.
2. Принципи організації багатопроцесорної обчислювальної системи.

Обчислювальні системи можуть будуватися на базі декількох комп'ютерів або на базі декількох процесорів. В першому випадку ОС буде *багатомашинною*, в другому – *багатопроцесорною*.

Багатомашинна ОС містить деяке число комп'ютерів, інформаційно взаємодіючих між собою. Комп'ютери можуть знаходитися поряд один з

одним, а можуть бути віддаленими один від одного на деяку, іноді значну відстань (обчислювальні мережі).

В **багатомашинних ОС** кожен комп'ютер працює під управлінням своєї операційної системи (ОпС). А оскільки обмін інформацією між комп'ютерами, що взаємодіють один з одним, виконується під управлінням ОпС, динамічні характеристики процедур обміну дещо погіршуються (потрібен час на узгодження роботи самих ОпС). Інформаційна взаємодія комп'ютерів в багатомашинній ОС може бути організовано на рівні:

- процесорів;
- оперативної пам'яті;
- каналів зв'язку.

При безпосередній взаємодії процесорів один з одним інформаційний зв'язок реалізується через регістри процесорної пам'яті і вимагає наявності в ОпС дуже складних спеціальних програм.

Взаємодія на рівні оперативної пам'яті (ОП) зводиться до програмної реалізації загального поля оперативної пам'яті, що дещо простіше, але також вимагає суттєвої модифікації ОпС. Під загальним полем мається на увазі рівнодоступність модулів пам'яті: всі модулі пам'яті доступні всім процесорам і каналам зв'язку.

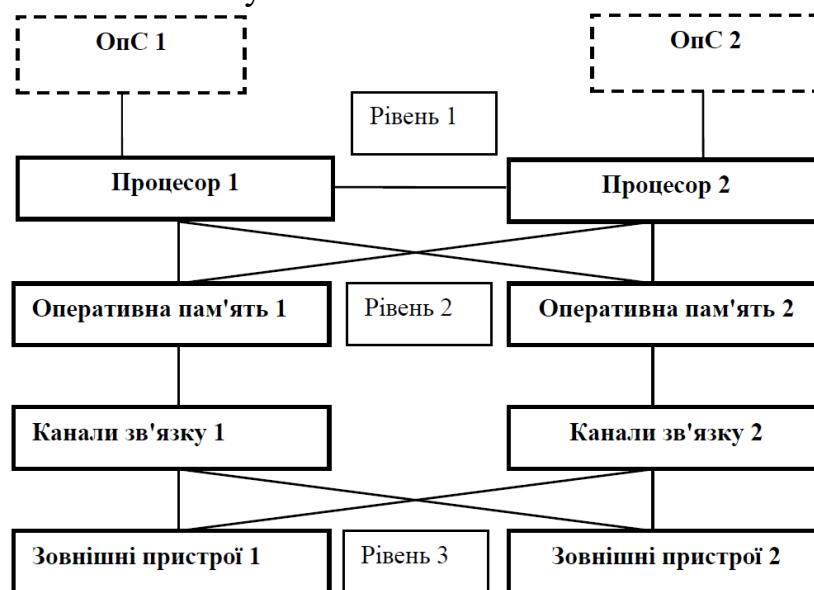


Рисунок 4.5 – Схема взаємодії комп'ютерів в ОС.

На рівні каналів зв'язку взаємодія організовується найбільш просто і може бути досягнуто зовнішніми по відношенню до ОпС програмами-драйверами, що забезпечують доступ від каналів зв'язку однієї машини до зовнішніх пристроїв інших (формується загальне поле зовнішньої пам'яті і загальний доступ до пристроїв вводу-виводу).

Зважаючи на складність організації інформаційної взаємодії на 1-у і 2-у рівнях в більшості багатомашинних ОС використовується 3-й рівень, хоча і динамічні характеристики (в першу чергу швидкодія), і показники надійності таких систем істотно нижче.

В *багатопроцесорній ОС* є декілька процесорів, інформаційно взаємопов'язаних між собою або на рівні регістрів процесорної пам'яті, або на рівні ОЗП. Цей тип взаємодії використовується в більшості випадків, бо організовується значно простіше і зводиться до створення загального поля оперативної пам'яті для всіх процесорів. Загальний доступ до зовнішньої пам'яті і пристроїв вводу-виводу забезпечується звичайно через канали ОЗП. Важливим є і те, що багатопроцесорна обчислювальна система працює під управлінням єдиної ОпС, загальної для всіх процесорів. Це істотно покращує динамічні характеристики ОС, але вимагає наявності спеціальної, дуже складної ОпС.

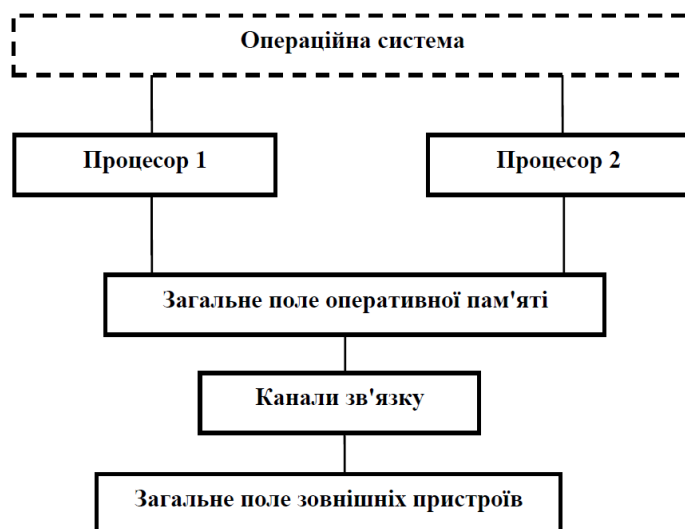


Рисунок 4.6 – Схема взаємодії процесорів в ОС.

Швидкодія і надійність багатопроцесорних ОС в порівнянні з багатомашинними, взаємодіючими на 3-у рівні, істотно підвищуються, *по – перше*, із-за більш швидкого обміну інформацією між процесорами, більш швидкого реагування на ситуації, що виникають в системі; *по – друге*, зважаючи на більший ступінь резервування пристроїв системи (система зберігає працездатність, поки працездатні хоча б по одному модулю кожного типу пристроїв).

Типовим прикладом масових багатомашинних ОС можуть служити комп'ютерні мережі, прикладом багатопроцесорних ОС - суперкомп'ютери.

Контрольні запитання:

1. Які особливості у функціонуванні векторної комп'ютерної системи?
2. Що відноситься до апаратних векторних засобів?
3. Які особливості у функціонуванні матричної комп'ютерної системи?
4. Які характеристики відрізняють SMP-системи?
5. Які особливості у функціонуванні MPP-систем?
6. Чим відрізняються багатомашинна обчислювальна система від
7. багатопроцесорної?

Тема 4.3 Ефективність обчислювальних систем.

Лекція 1 (2 год.) Показники ефективності обчислювальних машин.

План лекції:

1. Критерій ефективності обчислювальних машин.
2. Швидкодія обчислювальних машин.
3. Продуктивність обчислювальних машин.

Обчислювальну машину можна визначити множиною показників, що характеризують окремі її властивості. Виникає завдання введення міри для оцінки ступеня пристосованості ОМ до виконання покладених на неї функцій – міри ефективності.

Ефективність визначає ступінь відповідності ОМ своєму призначенню. Вона вимірюється або кількістю витрат, необхідних для отримання певного результату, або результатом, отриманим при певних витратах. Провести порівняльний аналіз ефективності декількох ОМ, ухвалити рішення на використання конкретної машини дозволяє критерій ефективності.

Критерій ефективності – це правило, що служить для порівняльної оцінки якості варіантів ОМ. Критерій ефективності можна назвати правилом переваги порівнюваних варіантів.

Будуються критерії ефективності на основі окремих показників ефективності (показників якості). Спосіб зв'язку між окремими показниками визначає вид критерію ефективності.

Як основні показники ОМ зазвичай розглядають: ємність пам'яті, швидкодію та продуктивність, вартість і надійність. Зупинимось тільки на показниках швидкодії та продуктивності, що зазвичай представляють основний інтерес для користувачів.

Швидкодія. Доцільно розглядати два види швидкодії: номінальну і середню.

Номінальна швидкодія характеризує можливості ОМ під час виконання стандартної операції. Як стандартну зазвичай вибирають коротку операцію додавання. Якщо позначити через $\tau_{\text{дод}}$ час додавання, то номінальна швидкодія визначиться з виразу

$$V_{\text{ном}} = \frac{1}{\tau_{\text{дод}}} \left[\frac{\text{оп}}{c} \right].$$

Середню швидкодію характеризує швидкість обчислень при виконанні еталонного алгоритму або деякого класу алгоритмів. Величина середньої швидкодії залежить як від параметрів ОМ, так і від параметрів алгоритму і визначається співвідношенням

$$V_{\text{сер}} = \frac{N}{T_e},$$

де T_e – час виконання еталонного алгоритму;

N – кількість операцій, що містяться в еталонному алгоритмі.

Позначимо через n_i число операцій i -го типу; l – кількість типів операцій в ($i = 1, 2, \dots, l$); τ_i – час виконання операції i -го типу.

Час виконання еталонного алгоритму розраховується за формулою

$$T_e = \sum_{i=1}^l \tau_i n_i.$$

Підставивши останнє співвідношення у вираз для $V_{сер}$, отримаємо

$$V_{сер} = \frac{N}{\sum_{i=1}^l \tau_i n_i}.$$

Звідки отримаємо

$$V_{сер} = \frac{1}{\sum_{i=1}^l \frac{n_i}{N} \tau_i}.$$

Позначивши частоту появи операції i -го типу в останньому виразу через $q_i = \frac{n_i}{N}$, запишемо остаточну формулу для розрахунку середньої швидкодії

$$V_{сер} = \frac{1}{\sum_{i=1}^l q_i \tau_i} \left[\frac{on}{c} \right]. \quad (1)$$

У виразі (6.6) вектор $\{\tau_1, \tau_2, \dots, \tau_l\}$ характеризує систему команд ОМ, а вектор $\{q_1, q_2, \dots, q_l\}$, що називається частотним вектором операцій, характеризує алгоритм.

Очевидно, що для ефективної реалізації алгоритму необхідно прагнути до збільшення $V_{сер}$. Якщо $V_{ном}$ головним чином відштовхується від швидкодії елементної бази, то $V_{сер}$ дуже сильно залежить від оптимальності вибору команд ОМ.

Формула (1) дозволяє визначити середню швидкодію машини під час реалізації одного алгоритму. Розглянемо більш загальний випадок, коли повний алгоритм складається з декількох окремих, періодично повторюваних алгоритмів. Середня швидкодія під час рішення повної задачі розраховується за формулою

$$V_{сер}^n = \frac{1}{\sum_{j=1}^m \sum_{i=1}^l \beta_j q_{ji} \tau_i} \left[\frac{on}{c} \right],$$

де m – кількість окремих алгоритмів;

β_j – частота появи операцій j -го окремого алгоритму в повному алгоритмі;

q_{ij} – частота операцій i -го типу в j -му окремому алгоритмі.

Позначимо через N_j і T_j – кількість операцій і період повторення j -го окремого алгоритму; $T_{\max} = \max_j (T_1, \dots, T_j, \dots, T_n)$ – період повторення повного

алгоритму; $a_j = T_{max}/T_j$ – циклічність включення j -го окремого алгоритму в повному алгоритмі.

Тоді за час T_{max} в ОМ буде виконано $N_{max} = \sum_{j=1}^m a_j N_j$ операцій, а частоту появи операцій j -го окремого алгоритму в повному алгоритмі можна визначити з виразу

$$\beta_j = \frac{\alpha_j N_j}{N_{max}}. \quad (2)$$

Для розрахунку за формулами (1, 2) необхідно знати параметри ОМ, представлені вектором $\{\tau_1, \tau_2, \dots, \tau_l\}$, параметри кожного j -го окремого алгоритму – вектор $\{q_{j1}, q_{j2}, \dots, q_{jl}\}$ і параметри повного алгоритму – вектор $\{\beta_1, \beta_2, \dots, \beta_m\}$.

Продуктивність ОМ оцінюється кількістю еталонних алгоритмів, що виконуються в одиницю часу:

$$P = \frac{1}{T_e} \left[\frac{\text{задач}}{c} \right].$$

Продуктивність під час виконання повного алгоритму оцінюється за формулою

$$P_n = \frac{1}{\sum_{j=1}^m \sum_{i=1}^l \alpha_j N_j q_{ij} \tau_i} \left[\frac{\text{задач}}{c} \right].$$

Продуктивність є більш універсальним показником, ніж середня швидкодія, оскільки в явному вигляді залежить від порядку проходження завдань через ОМ.

На практиці зазвичай використовують простіші вирази. Для оцінки часу виконання програми з динамічною кількістю команд N використовують вираз

$$T = \frac{N \cdot K}{F},$$

де K – середня кількість тактів, що витрачаються на вибірку і виконання однієї команди;

F – тактова частота процесора.

Для оцінки продуктивності використовують *пропускну здатність* процесора – кількість команд, що виконуються за одну секунду. У разі послідовного виконання команд пропускну здатність P_s визначається за формулою

$$P_s = \frac{F}{K}.$$

Для *конвеєрного* процесора пропускну здатність сама по собі ще не є свідомством хорошої продуктивності. Реальною мірою продуктивності комп'ютера є загальний час виконання програми. В загальному випадку n -

ступінчастий конвеєр потенційно підвищує продуктивність в n разів. Виходить, що чим більше значення n , тим вище продуктивність процесора. Проте реальна продуктивність конвеєрного процесора нижча. Кожного разу, коли відбувається зупинка конвеєра, потік команд, що обробляються процесором, різко скорочується. Таким чином, продуктивність конвеєра багато в чому залежить від таких чинників, як накладні витрати переходів і промахів у разі звернення до кеш-пам'яті.

Скорочення накладних витрат можна добитися за рахунок введення вторинного кеша, який розміщується між первинним кешем, інтегрованим у мікро- схему процесора, і основною пам'яттю. Крім того, в комп'ютерах використовується оптимізуючий компілятор, який по можливості віддаляє залежні один від одного команди, поміщаючи між ними інші. А також, якщо в процесорі існує черга команд, промах під час вибірки команди може мати значно менші негативні наслідки, оскільки процесор продовжує виконання команд, що знаходяться в черзі.

Для підвищення продуктивності конвеєрного процесора здавалося б доцільним розділяти процес виконання команд на якомога більшу кількість ступенів. Проте чим більше ступенів, тим вище вірогідність зупинок конвеєра. Це пов'язано з тим, що більшість команд виконуються паралельно, в зв'язку з чим навіть значно віддалені одна від одної команди, між якими є залежності, можуть викликати зупинки конвеєра, що, у свою чергу, приводить до зростання накладних витрат переходів. Через перераховані причини підвищення продуктивності за рахунок збільшення кількості ступенів виявляється не таким уже істотним.

Ще один важливий чинник, що впливає на продуктивність, – внутрішні затримки під час виконання процесором базових операцій. Особливої уваги заслуговує затримка в АЛП. У багатьох процесорах тактова частота підбирається так, щоб операція складання в АЛП здійснювалася за один такт. Решта операцій розділяється на кроки, що займають той же час, що і операція складання. При цьому АЛП може мати конвеєрну організацію.

У багатьох конвеєрних процесорах використовується від чотирьох до шести ступенів. У ряді процесорів процедура виконання команди ділиться на менші кроки, використовується більша кількість ступенів конвеєра і вища тактова частота. Наприклад, у процесорі UltraSPARC II застосовується 9-ступінчастий конвеєр, а в процесорі Intel Pentium Pro – 12-ступінчастий, Intel Pentium 4 містить 20-ступінчастий конвеєр і функціонує на тактовій частоті від 1,3 до 1,5 ГГц. Для прискорення роботи на кожному такті виконуються два ступені конвеєра.

Лекція 2 (2 год.) Продуктивність мультипроцесорних систем.

План лекції:

1. Оцінки продуктивності.
2. Оцінка програмної продуктивності.

3. Комунікаційні витрати.

Через особливості паралельних обчислень для оцінки їх ефективності використовують специфічну систему показників.

Число процесорів багатопроцесорної системи, що паралельно беруть участь у виконанні програми в кожен момент часу t , визначають поняттям *ступінь паралелізму* $D(t)$ (DOP, Degree Of Parallelism). Графічне зображення параметра $D(t)$ як функції часу називають *профілем паралелізму програми*.

Типовий профіль паралелізму для алгоритму декомпозиції (divide-and-conquer algorithm) показаний на рис. 1.

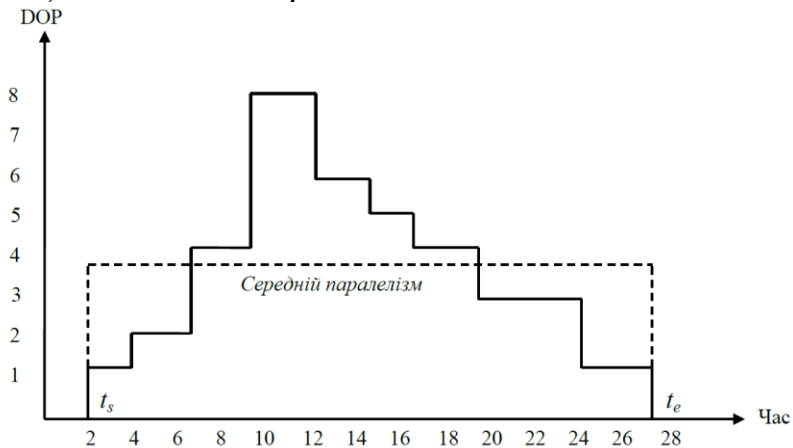


Рисунок 4.7 – Профіль паралелізму.

Зміни в рівні завантаження процесорів за час спостереження залежать від багатьох чинників (алгоритму, доступних ресурсів, ступеня оптимізації, що забезпечується компілятором, і т.д.).

Надалі виходитимемо з наступних припущень: система складається з n гомогенних процесорів; максимальний паралелізм у профілі рівний m і, в ідеальному випадку, $n \gg m$. Продуктивність E одиничного процесора системи виражається в одиницях швидкості обчислень (кількість операцій в одиницю часу) і не враховує витрат, пов'язаних із зверненням до пам'яті і пересилкою даних. Якщо за спостережуваній період завантажені i процесорів, то $D = i$.

Загальний об'єм обчислювальної роботи W (команд або обчислень), виконаної починаючи із стартового моменту t_s до моменту завершення t_e , пропорційний площі під кривою профілю паралелізму:

Середній паралелізм A визначається як

$$A = \frac{1}{t_e - t_s} \int_{t_s}^{t_e} D(t) dt .$$

У дискретній формі це можна записати так:

$$A = \frac{\sum_{i=1}^m i \cdot t_i}{\sum_{i=1}^m t_i}.$$

Прискорення (speedup) або, точніше, середнє прискорення за рахунок паралельного виконання програми – це відношення часу, потрібного для виконання якнайкращого з послідовних алгоритмів на одному процесорі, і часу паралельного обчислення на n процесорах.

Без урахування комунікаційних витрат прискорення $S(n)$ визначається як

$$S(n) = \frac{T(1)}{T(n)}.$$

Як правило, прискорення задовольняє умову $S(n) \leq n$.

Ефективність (efficiency) n -процесорної системи – це прискорення на один процесор, що визначається виразом

$$E(n) = \frac{S(n)}{n} = \frac{T(1)}{n \cdot T(n)}.$$

Ефективність зазвичай відповідає умові $1/n \leq E(n) \leq n$.

Залежно від кількості процесорів у системі розрізняють три можливі варіанти прискорень.

Власне прискорення визначається шляхом реалізації паралельного алгоритму на одному процесорі.

Якщо прискорення, досягнуте на n процесорах, дорівнює n , то говорять, що алгоритм показує *лінійне прискорення*.

У виняткових ситуаціях прискорення $S(n)$ може бути більше, ніж n . У цих випадках іноді застосовують термін *суперлінійне прискорення*.

До чинників, що обмежують прискорення, слід віднести:

Програмні витрати. Навіть якщо послідовні і паралельні алгоритми виконують одні і ті ж обчислення, паралельним алгоритмам властиві додаткові програмні витрати – додаткові індексні обчислення, що неминуче виникають через декомпозицію даних і розподіл їх по процесорах; різні види облікових операцій, потрібні в паралельних алгоритмах, але відсутні в алгоритмах послідовних.

Витрати через дисбаланс завантаження процесорів. Між точками синхронізації кожен з процесорів повинен бути завантажений однаковим об'ємом роботи, інакше частина процесорів чекатиме, поки останні завершать свої операції. Ця ситуація відома як дисбаланс завантаження. Таким чином, прискорення обмежується найбільш повільним з процесорів.

Комунікаційні витрати. Якщо прийняти, що обмін інформацією та обчислення можуть перекриватися, то будь-які комунікації між процесорами знижують прискорення. В плані комунікаційних витрат важливий рівень гранулярності, що визначає об'єм обчислювальної роботи, яка виконується між комунікаційними фазами алгоритму. Для зменшення комунікаційних

витрат вигідніше, щоб обчислювальні гранули були достатньо великими і частка комунікацій була менша.

Контрольні запитання:

1. Що таке *номінальна* та *середня швидкодія* обчислювальної машини?
2. Як оцінюється продуктивність обчислювальної машини?
3. Як оцінюють прискорення та ефективність комп'ютерної системи?
4. Як оцінюють програмні витрати?
5. Який вплив комунікаційних витрат на ефективність комп'ютерної системи?

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Мельник А. О. Архітектура комп'ютера: наукове видання. Луцьк: Волинська обласна друкарня, 2008. 470 с.
2. Тарарака В. Д. Архітектура комп'ютерних систем: навч. посіб. / Житомир : ЖДТУ, 2018. 383 с.
3. Карачка А. Ф., Дудко О. І. Архітектура комп'ютерів. навч. посіб. / за ред. А. О. Саченка. Тернопіль: Економічна думка, 2009. с. 422.
4. Комп'ютери та комп'ютерні технології: навч. посіб. / Ю. Б. Бродський, К. В. Молодецька, О. Б. Борисюк, І. Ю. Гринчук. – Житомир : Вид-во «Житомирський національний агроекологічний університет», 2016. – 186 с.
5. Матвієнко М. П. Архітектура комп'ютера: навч. посіб. для студ. вищ. навч. закл. / М. П. Матвієнко, В. П. Розен, О. М. Закладний. – К. : Ліра, 2013. – 264 с