МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМПЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра комп'ютерної інженерії

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни "Комп'ютерні системи"

для студентів спеціальності 123 – «Комп'ютерна інженерія»

Тернопіль
2022

Дубчак Л.О. Конспект лекцій з дисципліни “Комп’ютерні системи” для студентів спеціальності 123 – «Комп’ютерна інженерія» – Тернопіль, 2022. – 64 с.

Укладачі: Дубчак Л.О., к.т.н., доцент

Рецензенти: Якименко І.З., к.т.н., доцент, доцент кафедри кібербезпеки ЗУНУ
Васильків Н.М., к.т.н., доцент кафедри ІОСУ ЗУНУ

Відповідальний за випуск: Березький О.М., д.т.н., професор

Конспект лекцій з дисципліни “Комп’ютерні системи” для студентів спеціальності 123 – «Комп’ютерна інженерія»” містить теоретичні основи побудови та проектування комп’ютерних систем.

Методичні рекомендації розглянуто та рекомендовано до друку на засіданні кафедри комп’ютерної інженерії протокол № 11 від 16.06.2022 р.

1. ЕЛЕМЕНТИ ТЕОРІЇ СИСТЕМ

1.1 Поняття системи

В залежності від широти представлення прийнято декілька визначень системи.

Найбільш загальне: ***Система – це сукупність об'єктів, на які накладені зв'язки***. Причому під поняттям «*зв'язок*» розуміються такі терміни як: взаємне розташування об'єктів, організація, сполучення, взаємодія, кількісне співвідношення та. ін.

Більш конкретне: ***Система – це об'єкт будь-якої природи чи сукупність об'єктів довільної, в тому числі і різної природи, який володіє вираженою системною властивістю, тобто властивістю, якою не володіє ні одна з частин системи при будь-якому способі поділу, і яка не виводиться з властивостей частин.***

Найбільш важливим наслідком цього визначення є те, що властивість системи формується в процесі певної організації її складових, тобто вона залежить як від складових системи так і накладених на них зв'язків.

Наглядними прикладами формування системних властивостей є такі системи як слово, число, речення.

Дійсно, слово складається з букв, які є символами, що відображають звуки. Слово ж володіє семантичною (змістовною) властивістю, яка формується шляхом організації букв певним чином. Так з одних і тих-же елементів (букв) за рахунок накладання зв'язків (розташуванні у слові) можна сформувати різні системи (слова). Слід вважати що система може складатися з одного елемента, однак властивості системи і елемента можуть бути різні. Так такі слова як „я", „і", „а" і т. п. по позначенню співпадають з відповідними буквами, однак властивості у них різні.

У прийнятій порядковій системі числення властивість числа (його кількісний еквівалент) залежить від порядку розташування (зв'язків) цифр (елементів системи). Не слід плутати поняття цифра і число (аналогічно з буквою і словом) , так як вони володіють зовсім іншими властивостями. Так цифра (0, 1, 2, ..., 9) є символ для представлення чисел, тому число „8" і цифра „8" хоч і співпадають по запису мають зовсім інші властивості. Тому беззмістовними є висловлювання типу „сума двох цифр ..." і т. п., хоч, нажаль, це часто зустрічається навіть у авторитетних виданнях.

Аналогічно можна сформувати системні властивості речення, параграфу, глави (розділу), статті, книги.

Прикладами формування властивості інших систем:

- властивості фізичних елементів таблиці Мендєлєєва різні за рахунок організації одних і тих же нуклонів у ядрі атома;
- властивості вуглеводів, так бензини, спірти, пластики і т. п. мають однаковий елементний склад, однак, за рахунок різноманітності зв'язків, різні властивості;
- властивості електронної апаратури, формуються за рахунок з'єднань типових електронних елементів згідно певної схеми;
- в будівництві, з одних і тих же типових елементів формуються будови різного призначення;
- організаційні системи, з однієї елементної бази (людей) за рахунок організації формуються різні структури (держава, парламент, армія, міністерство, адміністрація, відділ, академічна група студентів, сім'я і т. п.).

1.2. Характеристики систем

Будь-яка система характеризується трьома параметрами (Рис.1.1):

- множиною впливів на систему зовнішнього середовища (зовнішнім впливом, входом системи, вхідною дією) – U.
- множиною змінних стану системи (станом системи) – X;
- множиною впливів системи на зовнішнє середовище (вихід системи, вихідна дія, реакція сисчтеми) – Y.

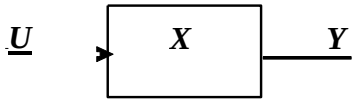


Рисунок 1.1 – Характеристики системи

Фактично сприйняття любого об'єкту, виділення його серед інших (ідентифікація) здійснюється за його реакцією на зовнішній вплив і/або на внутрішній стан. Дійсно у абсолютній темряві, використовуючи тільки органи зору, розпізнати об'єкт неможливо, при умові що він не володіє внутрішнім джерелом випромінювання.

Прийняття цих характеристик обумовлено основною постановочною задачею – навіщо нам потрібно досліджувати системи? Робота заради роботи не робиться. Причому під терміном дослідження розуміються поняття: дослідження, вивчення, розробка, проектування і т. п. Відповіддю є „Щоб використовувати системи для власних цілей (потреб).

Якщо проаналізувати процес використання систем то можна зробити висновок, що він полягає у цілеспрямованому впливі на систему з метою отримання необхідних результатів (виходу системи), інакше кажучи у керуванні системою. Однак, щоб бути спроможнім здійснювати таке керування, треба знати який вплив треба здійснити на систему, щоб отримати заданий результат. Формально це можна описати за допомогою операторів відображення:

$$L : (U, X) \to Y, \quad (1.1)$$

вхід, стан у вихід;

$$G : (U, X) \to X, \quad (1.2)$$

вхід, стан у стан.

У повсякденному житті ми весь час використовуємо знання, які формально представляються цими операторами, навіть не задумуючись про це. Як правило це представлено у вигляді інтуїції і навичок, що формуються у людини з дня її народження. Так пишучі олівцем, або ручкою ми інтуїтивно знаємо як треба вплинути на систему, щоб досягти необхідного результату, тобто знаємо закономірність (1.1).

При використанні складних технічних засобів необхідна інформація у вигляді оператора (1.1), а при їх налагоджуванні, або ремонту і (1.2), як правило, отримується з технічної документації, зокрема з інструкції по експлуатації.

Таким чином кінцевою метою дослідження систем є отримання закономірностей типу (1.1) і (1.2) у вигляді необхідному при даному розгляді (описовому, причиново-наслідковому, кількісному і т. п.). Таке представлення системи називають моделлю системи, а процес знаходження цих закономірностей і їх дослідження моделюванням.

1.3. Класифікація представлення систем

Класифікація (таксономія) систем теж є деяка абстрактна система об'єктами якої є властивості на які накладені зв'язки.

Часто вживане поняття „класифікація систем" є не зовсім вдалий, так як одна і таж система в залежності від мети дослідження і глибини розгляду може відноситись до різних класів. Фактично здійснюється класифікація з точки зору абстрактного представлення систем і тому краще застосовувати поняття „класифікація представлення систем"

По взаємодії з оточуючим середовищем представлення систем можна поділити на

- ***Відкриті,*** взаємодіють з оточенням.
- ***Замкнуті,*** не взаємодіють з оточенням.

Очевидно, що в природі замкнутих систем не існує (без зовнішньої взаємодії їх неможливо спостерігати, їх нема). Однак, при деякому розгляді, для спрощення опису - системи можна вважати замкнутими.

По інерційності представлення систем можна поділити на:

- ***Статичні*** (без інерційні), вважається, що переходи можуть відбуватися миттєво, процеси в них описуються за допомогою функціональних залежностей між входом станом і виходом, тобто знаючи, у конкретний момент часу, значення входу і стану системи можна статистично визначити значення виходу.
- ***Динамічні*** (інерційні), вважається, що переходи не можуть відбуватися миттєво, процеси в них описуються за допомогою інтегродиференційних рівнянь.

Очевидно, що в природі статичних систем не існує, навіть електрон має кінцеву масу, а миттєвій зміні стану відповідає нескінченне прискорення, тобто згідно другому закону Ньютона $a=F/m$ для цього необхідна нескінченна сила.

Однак у випадках коли перехідними процесами можна нехтувати систему можна розглядати як статичну.

По визначеності представлення систем можна поділити на:

- ***Детерміновані (визначені)***, вважається, що невизначеність повністю відсутня і при наявності повної інформації про функції входу і стану вихід системи визначається однозначно. S1 – системи.
- ***Стохастичні (ймовірностні)***, вважається, що процеси в системі мають статистичний характер, описуються за допомогою апаратів теорії ймовірності та математичної статистики. S2 – системи.
- ***Хаотичні***, повністю невизначені, вважається, що зовсім не можна передбачити поведінку системи, прикладом є Броунівський рух. S3 – системи.
- ***Складні*** системи, володіють особливими властивостями (унікальності, слабо передбачуваності, негентропійності), прикладом є складні технічні системи, біологічні, організаційні системи. Більш детально будуть розглянуті далі. S0 – системи.

1.4. "Кортежний" опис систем

З іншого боку систему можна розглядати, як сукупність елементів, яка володіє наступними ознаками:

1) зв’язками, які дозволяють через переходи по них від елемента до елемента з’єднати два будь-яких елемента сукупності (зв’язність системи);
2) властивістю (призначенням, функції), відмінною від властивостей окремих елементів сукупності (функція систем).

Виходячи з цього визначення найбільш простим є "кортежне" представлення системи:

$$\Sigma: \{\{M\}, \{u\}, F\}, \qquad (1.3)$$

де Σ – система, $\{M\}$ – сукупність елементів; $\{u\}$ – сукупність зв’язків; F – функція системи. Запис (1.3) розглядається як найбільш простий опис змісту системи.

Практично будь-який об’єкт з певної точки зору може розглядатися як система. Важливо усвідомити – чи корисний такий погляд, чи розумно рахувати даний об’єкт елементом. Так, системою можна рахувати радіотехнічну плату, що перетворює вхідний сигнал у вихідний. Для спеціаліста з елементної бази системою буде слюдяний конденсатор на цій платі, а для геолога – і сама слюда, що має досить складну будову.

Великою системою називається система, що включає значну кількість однотипних елементів і однотипних зв'язків.

Різниця між системою, великою системою і складною системою умовна. Так, корпуси ракет чи кораблів, які на перший погляд однорідні, звичайно відносять до складної системи – через наявність перегородок різного виду, підсилювачів, шарової конструкції.

Важливим класом складних систем є так звані автоматизовані системи. Слово "автоматизований" вказує на присутність людини, використання її активності всередині системи при збереженні значної ролі технічних засобів. Для складної системи автоматизованому режиму надається перевага. Наприклад, приземлення літака виконується при участі людини, а автопілот звичайно використовується лише при відносно простих рухах. Також типова ситуація, коли рішення, що виробляється технічними засобами, затверджується до виконання людиною.

Тому автоматизованою системою називається складна система з визначеною роллю елементів двох типів: – технічних засобів; – дій людини. Її символьний запис (можна порівняти з (1.3):

$$\Sigma^{A}: \{\{M^{T}\}, \{M^{Л}\}, \{M'\}, \{u\}, F\}, \quad (1.4)$$

де МТ – технічні засоби, в першу чергу ЕОМ; МЛ – рішення та інша активність людини; М' – решта елементів системи.

1.5. Структура та ієрархія

Структурою системи називається її розбиття на групи елементів із вказуванням зв'язків між ними, яка не змінюється за весь час розгляду і дає уявлення про систему в цілому.

Розбиття може мати матеріальну (речову), функціональну, алгоритмічну та іншу основу. Групи елементів в структурі звичайно виділяються по принципу простих або відносно більш слабких зв'язків між елементами різних груп. Структуру системи зручно представляти у вигляді графічної схеми, що складається з модулів (груп) і ліній (зв'язків), які з'єднують їх. Такі схеми називаються структурними.

Для символьного запису структури введемо замість сукупності елементів {М} сукупність груп елементів {М^} і сукупності зв'язків між цими групами {u^}. Тоді структура системи може бути записана як

$$\Sigma\,\Sigma: \{\{M^{}\}, \{u^{}\}\}. \quad (1.5)$$

Структуру (1.5) можна отримати з (1.3) об'єднанням елементів в групи. Відмітимо, що функція (призначення) F системи в (1.5) опущена, оскільки структура може бути в визначеному степені безвідносна до неї. Відмітимо також, що елементи М з якої-небудь групи М^ будуть, як правило, неоднорідні.

Наприклад, матеріальна структура збірного мосту складається з його окремих секцій, які збираються на місці. Структурна схема такої системи вкаже тільки ці секції і порядок їх з'єднання. Останній і є зв'язками, які тут носять силовий характер. Приклад функціональної структури – це ділення двигуна внутрішнього згорання на системи живлення, змащування, охолодження, передачі силового моменту та ін. Приклад системи, де матеріальні і функціональні структури об'єднані це відділи проектного інституту, який займається різними сторонами однієї і тієї ж проблеми. Типовою алгоритмічною структурою буде алгоритм (схема) програмного засобу, яка вказує послідовність дій. Також алгоритмічною структурою буде інструкція, яка визначає дії при пошуку неполадок технічного об'єкту.

Прикладами структур таких типів є календар (часова структура) чи ділення книжки на розділи. Ситуація з книгою цікава тим, що тут основа ділення може бути інформаційною (в науковій літературі), речовою (для типографії розділ – це кількість паперу і праці робітників) чи більш складною, наприклад, побудованою на наборі естетичних впливів на читача (для художньої літератури).

Структура системи може бути охарактеризована типами зв'язків, які існують (чи переважають) в ній. Найпростішими з них є послідовне, паралельне з'єднання елементів і зворотній зв'язок (рисунок 1.2).

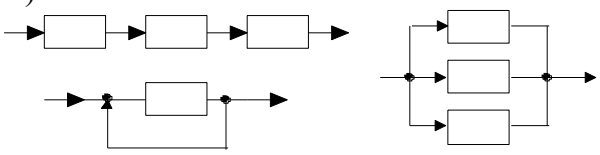


Рисунок 1.2 – Найпростіші типи зв'язків

Декомпозицією називається поділ системи на частини, зручний для яких-небудь операцій з цією системою. Декомпозиція, як правило, здійснюється за принципом слабкості зв'язків, тобто зв'язки між елементами підсистеми набагато сильніші ніж між елементами різних підсистем. Прикладами декомпозиції будуть: розгляд фізичного явища, чи математичний опис окремо для даної частини системи; поділ об'єкту на окремі зони обслуговування, частини, які проектуються окремо.

Важливим стимулом і суттю декомпозиції є спрощення системи, надто складної для розгляду в цілому. Таке спрощення може:

а) фактично приводити до заміни системи на деяку іншу, яка в якому-

небудь змісті відповідає вихідній – як правило, це робиться шляхом введення гіпотез про відкидання чи ослаблення окремих зв'язків в системі;

б) повністю відповідати вихідній системі і при цьому полегшувати роботу з нею – така декомпозиція, яка називається строгою, потребує спеціальних процедур узгодження і координації розгляду частин.

Ієрархією називають структуру з наявністю підлеглості, тобто нерівноправних зв'язків між елементами, коли вплив в одному з напрямків здійснює набагато більшу дію на елемент, ніж іншому.

Види ієрархічних структур різноманітні. Серед них зустрічаються такі, як кільцеві (перший елемент домінує над другим, другий – над третім і т.д., але останній – над першим) чи домінування, які змінюють напрямок. Але основних, важливих для практики ієрархічних структур всього дві – деревоподібна і ромбоподібна (рисунок 1.3).

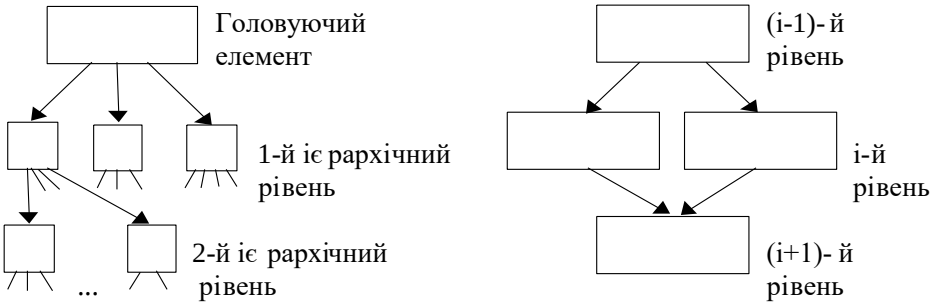


Рисунок 1.3 - Приклади ієрархічних структур
а - деревоподібна; б - ромбоподібна

Деревоподібна структура найбільш проста для аналізу і реалізації. В ній майже завжди зручно виділяти так звані ієрархічні рівні – групи елементів, які знаходяться на однаковому (по числу проміжних елементів) віддаленні від верхнього (головуючого) елемента.

Приклади таких структур в штучних і живих системах надзвичайно багато чисельні. Ними будуть: а) ланцюг "міністерство – главк – завод – цех – бригада – ланка"; б) задача проектування технічного об'єкту – від його загальних характеристик (верхній рівень) через

проектування основних частин, функціональних систем, груп агрегатів, механізмів до рівня окремих деталей; в) ієрархія цілей в задачі автоматизованого виробництва – від мети ділянки , яка полягає у максимальному випуску продукції, до програмного забезпечення окремої операції на станку (ціль – операція); г) в живій природі – ієрархія за ознакою керованості процесів в організмі, ієрархія в стаді та ін.

Ромбоподібна структура веде до подвійної (деколи і більше) підлеглості, звітності, приналежності нижнього елементу. В техніці – це участь даного елементу у роботі більш ніж одного вузла, блока, використання одних і тих же даних чи результатів вимірювань в різних задачах.

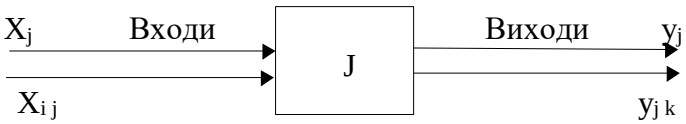
Будь-яка ієрархія, в принципі, звужує можливості і особливо гнучкість системи. Елементи нижнього рівня сковуються домінуванням зверху, вони здатні впливати на це домінування (управління) лише частково і, як правило, із затримкою. Однак введення ієрархії різко спрощує створення і функціонування системи, і тому його можна рахувати вимушеним, але необхідним прийомом розгляду складних технічних систем. Недарма той чи інший ступінь ієрархії спостерігається в переважаючій більшості реальних систем.

1.6. Модульний поділ систем

Група елементів системи, яка описується тільки своїми входами і виходами і яка володіє певною цілісністю, називається модулем.

Система може представлятися набором модулів і сама розглядатися як модуль. Модульна побудова системи, як правило, визначає її декомпозицію. Нерідко вона визначає і структуру. Однак значення поняття модуля в системному аналізі і суміжних з ним дисциплінах ще ширше. Ділення системи на модулі – це зручний і найбільш розповсюджений прийом роботи з штучними системами, включаючи їх створення (проектування), перевірку, налагоджування, вдосконалення. Саме модульна побудова системи разом з принципом введення все більш великих модулів при збереженні об'єму входів і виходів, який спостерігається, дозволяє розглядати різної системи складності. Прикладами реалізації цього положення на практиці є створення з сотень тисяч елементів (матеріальних, інформаційних, енергетичних) обчислювальних машин, а також створення інформаційних систем і обчислювальних мереж, які охоплюють цілий ряд країн, включаючи їх багаторівневе програмне забезпечення. Розробка таких систем звичайно йде "зверху", з продумуванням призначення, входів і виходів модулів верхнього рівня, і далі спускається вниз, все в більшій степені деталізуючи систему.

Схематичне зображення модуля наведено на рис. 1.4.



Модуль

Рисунок 1.4 - Модуль J

Тут $\boldsymbol{X_j}$ – зовнішні (від "не-системи") впливи на елементи модуля $\boldsymbol{J}$; $\boldsymbol{X_{ij}}$ – зв'язки від інших елементів системи на елементи модуля $\boldsymbol{J}$; $\boldsymbol{Y_{jk}}$ – зв'язки (впливи) від елементів модуля J на інші елементи системи; $\boldsymbol{Y_j}$ – зв'язки (впливи)від елементів модуля на не-систему, їх також можна розглядати як частину $\boldsymbol{F_j}$ функції системи $\boldsymbol{F}$, яка реалізується модулем $\boldsymbol{J}$. В цьому випадку маємо $\{\boldsymbol{Y_j}\}=\boldsymbol{F_j}$. Тапер можна представити модуль у вигляді перетворення

$$(\{Uj\}, \{Uij\}, J) \rightarrow (\{yj\}, \{yij\}). \qquad (1.6)$$

Поняття модуля близьке до концепції "чорного ящика" в кібернетиці (об'єкт, в якому відома тільки залежність виходів від входів). Однак на відміну від такої крайньої ситуації, при

дослідженні складних систем, є можливість аналізу того, що відбувається всередині модуля, але зручно не робити цього на певній стадії розгляду.

Важливість понять модуля, входу, виходу підкреслюється великою кількістю їх синонімів в різних розділах науки та техніки. Так, наприклад, синонімами модуля є “агрегат”, “блок”, “вузол”, “механізм” в техніці; “підпрограма”, “програмний модуль”, “логічний блок” – в програмуванні; “підрозділ”, “комісія” – в організації та управлінні. Типовими входами і виходами є пари “сигнал – відгук”, “вплив (дія) – реакція”, “запит – відповідь”, “аргумент – рішення”, або, більш ширше, “інформація – прийняття рішення”, “управління – рух” тощо.

1.7. Процеси у системах

Введемо поняття стану і процесу в системі. Для цього спочатку розглянемо деякий виділений елемент. Він може бути внесений в систему, виключений із системи, перенесений в ній з одного місця в інше. Крім того, можуть бути змінені його зв’язки. Всі ці ситуації відносяться до зміни структури системи.

Але можливі перетворення іншого роду. Будь-який елемент володіє рядом властивостей, характеристик, які також можуть змінюватись в процесі розгляду системи. Внаслідок цього можуть змінитися властивості, характеристики групи елементів, модуля та системи в цілому.

Зафіксуємо всі значення характеристик в системі, важливих для цілей розгляду. Таку ситуацію назвемо станом системи.

Нехай тепер хоча б одна така характеристика змінилась. Це буде новий стан системи. Аналогічно можна розглянути третій – і т.д. – стани, тобто їх набір. Але набір станів – це ще не процес. Нехай вибраний деякий фізичний параметр (частіше всього час) – такий, що різні стани відповідають різним його значенням.

Процесом назвемо набір станів системи, що відповідає впорядкованій неперервній чи дискретній зміні деякого параметру, що визначає характеристики (властивості) системи.

Для пояснення наведемо приклад.

Стан робота-маніпулятора будемо характеризувати положенням його основного робочого органу – захвату. Зробимо серію фотографій маніпулятора, на яких захват буде знаходитись в різних точках простору. Чи буде цей набір фотознімків характеризувати який-небудь процес? Без допоміжної інформації це невідомо. Якщо це послідовні в часі положення захвату, то – так (параметр процесу - час). Якщо ж знімки зроблені навгад чи перемішені, то відповідний набір стану не буде процесом.

Процес руху (зміни) системи в часі називають динамікою системи. Параметрами процесу можуть також виступати температура, тиск, інші фізичні величини. В якості параметра іноді виступають лінійні і кутові координати (наприклад: процес зміни тиску з висотою) і навіть швидкості. Однак більш типове відношення цих величин до характеристик системи, які самі залежать, наприклад, від часу.

Процеси в системі можуть відігравати окрему роль. Так, в системі автоматизованого проектування процес проектування як рух від технічного завдання до робочих креслень є основною функцією системи. І в цілому функціонування (а також створення) складної системи звичайно є процесом. Але в тому ж проектуванні напевно необхідно враховувати цілий ряд внутрішніх процесів: якщо щось рухається, то рівняння руху, якщо йдуть хімічні перетворення, то хід реакції. Таким чином, типовий облік процесів в системі як спосіб отримання залежностей виходів від входів в модулях різних ієрархічних рівнів. При цьому, в принципі, неважливо, чи сприяє в цілому даний процес виконанню системою її функції чи перешкоджає цьому. До останнього відносяться, наприклад, процеси зносу, старіння, а також дії протилежної сторони в ігрових ситуаціях.

1.8. Основні задачі дослідження систем (аналіз і синтез)

Задача синтезу – полягає в знаходженні структури і основних параметрів системи по заданих властивостях системи.

Задача аналізу – по відомій структурі і відомим параметрам вивчається поведінка системи, тобто досліджуються її властивості і характеристики.

Синтез – це процес породження функцій і структур, необхідних і достатніх для отримання визначених результатів.

Абстрактний синтез – це синтез функції, оскільки виявляючи функції, які реалізуються системою, визначають деяку абстрактну систему, про яку відомо тільки те, що вона буде робити.

Структурний синтез – породження структури, яка реалізує задані функції.

Вихідною інформацією в задачах синтезу є наступні відомості:

- функція системи – як правило представляється переліком задач, розв'язок яких покладено на систему;
- перелік обмежень на характеристики системи (часове обмеження, наприклад);
- критерій ефективності, який встановлює спосіб оцінки якості системи в цілому.

Аналіз – процес визначення властивостей, притаманних ситемі.

Типова задача аналізу - відомі функції і характеристики елементів системи (підсистем) і визначена її структура; необхідно визначити функції або характеристики самої системи як сукупності.

Показники, які характеризують властивості системи, можуть бути визначені:

1) Шляхом обробки результатів натурного експерименту.
2) В результаті фізичного або математичного моделювання процесів функціонування системи.

Проведення натурного експерименту доцільне при виконанні наступних умов:

1) Система може функціонувати в режимах, які допускають досягнення цілі експерименту.
2) Є можливість фіксації всієї необхідної інформації без суттєвих затрат.
3) Фіксація і обробка інформації в реальному масштабі часу задовільняють поставленим термінам експерименту.

У випадку складних систем ці умови часто не виконуються, тому найбільш ефективним засобом аналізу складних систем є математичне моделювання.

Задачі аналізу включають три етапи:

1) необхідно визначити причинно-наслідкові зв'язки, притаманні об'єкту, що аналізується, і побудувати концептуальну (причинно-наслідкову) модель об'єкта, яка виявляє сутність процесів, які відбуваються в системі. При побудові концептуальної моделі встановлюють наявність залежності між характеристиками процесів і параметрами об'єкта, які необхідно використати в моделюванні;
2) на базі концептуальної моделі будується математична модель, яка виявляє кількісні співвідношення між характеристиками і параметрами. Дослідження цих співвідношень дозволяють виявити властивості об'єкта, граничні і екстремальні значення характеристик, взаємозв'язки між ними;
3) перевірка достовірності моделі і отримання на її базі теоретичних результатів. Це здійснюється шляхом співставлення модельних залежностей з даними експерименту або даними, отриманими на основі інших моделей.

На відміну від задач аналізу, де характеристики процесів визначені як функції параметрів системи, при синтезі розв'язується задача вибору параметрів системи, при яких

задовольняються задані вимоги до характеристик процесів, тобто розв'язок задачі синтезу зводиться до оптимізації системи по заданому критерію ефективності з врахуванням обмежень на характеристики і параметри.

1.9. Методика формалізації об'єкта

В практичних задачах, на відміну від задач математичних, не завжди буває ясно, що задано і що необхідно довести. Як правило, задається реальний об'єкт (явище природи, фізичний, біологічний або виробничий процес).

Основні етапи побудови і аналізу реальних об'єктів подано на рисунку 1.6.

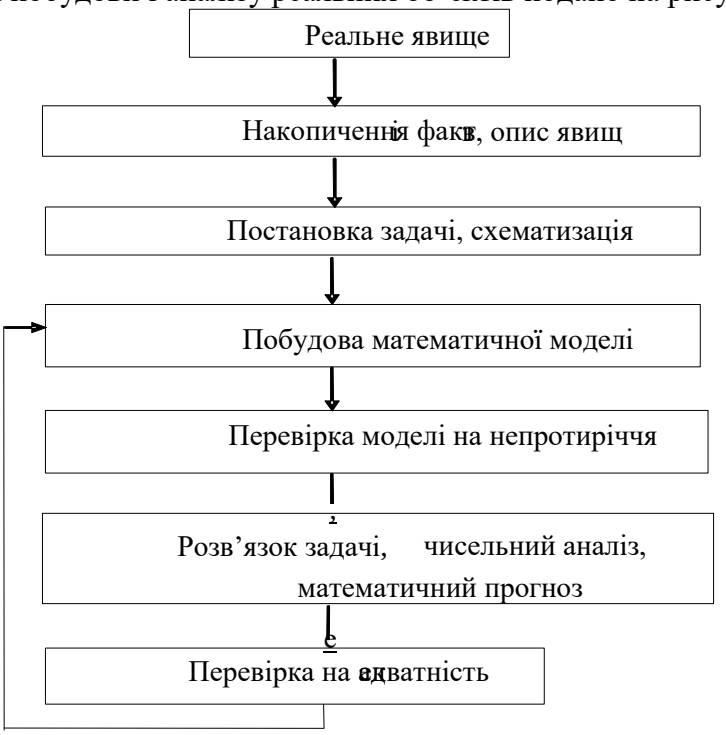


Рисунок 1.6 – Основні етапи побудови і аналізу моделей

Процес формалізації об'єкта складається з трьох етапів:

- побудова змістовного опису реального процесу (об'єкта) – дескриптивна модель;
- побудова формалізованої схеми (присутнє не завжди);
- розробка математичної моделі.

Дескриптивна модель – словесний опис закономірностей, які характеризують процес, який досліджується, а також постановка прикладної задачі або чітке формулювання мети досліджень (як правило складається спеціалістом в конкретній області без активної участі системотехніка).

Обов'язково повинна містити перелік залежностей, які підлягають оцінці.

Формалізована схема здійснюється, якщо неможливий безпосередній перехід від дескриптивної моделі до математичної. Може бути словесною, але містити строгий формальний опис об'єкта. Тут вибирається сукупність характеристик станів і параметрів об'єкта, формується математична мета досліджень. Вихідна інформація представляється у вигляді графіків, таблиць.

2. КЛАСИФІКАЦІЯ КОМП'ЮТЕРНИХ СИСТЕМ

2.1 Обгрунтування використання класифікації (таксономія)

Довільна схема класифікації повинна володіти наступними якостями: «можливістю класифікації всіх, як існуючих, так перспективних обчислювальних архітектур;

- диференціювання істотно відмінних обчислювальних архітектур;
- однозначність трактування довільних ЗОТ (засобів обчислювальної техніки) та їх архітектур.

Назви, що характеризують лише загальні принципи функціонування ЗОТ:

- векторно-конвеєрні;
- масивні-рівнобіжні (паралельні);
- комп'ютери із широким командним словом;
- систолічні масиви;
- гіперкуби;
- спецпроцесори і мультипроцесори;
- ієрархічні і кластерні комп'ютери;
- dataflow;
- матричні ЕОМ,
- інші.

Параметри ЗОТ, як, наприклад:

- організація пам'яті;
- топологія зв'язку між процесорами;
- синхронність роботи окремих процесорів чи пристроїв;
- спосіб виконання арифметичних операцій, то число різних архітектур стане і зовсім неозорим.

Основа класифікації вирішується у залежності від того, на вирішення якої задачі вона спрямована.

Класифікація повинна допомогти користувачу (оператору, програмісту) розібратися

- що являє собою кожна архітектура;
- взаємозалежність її складових між собою;
- що необхідно враховувати для написання ефективних програм;
- на який клас архітектур варто орієнтуватися для рішення необхідного класу задач.

Вдала класифікація в стані підказати розробнику можливі шляхи удосконалювання комп'ютерів і в цьому змісті вона повинна бути досить змістовною. Тому слід вибрати класифікації, у яких в якості критеріїв диференційовані принципи функціонування чи параметри ЗОТ або ж введені нові істотні поняття.

2.2 Доповнення Ванга і Бріггса до класифікації Флінна

Базові класи (SISD, SIMD, MIMD) доповнюються згідно наступних ознак:

Клас **SISD** розбивається на два підкласи (рис.2.1):

- архітектури з **єдиним функціональним пристроєм,** наприклад, PDP-11;
- архітектури, що мають у своєму складі **кілька функціональних пристроїв** - CDC 6600, CRAY-1, FPS AP-120B, CDC Cyber 205, FACOM VP-200.

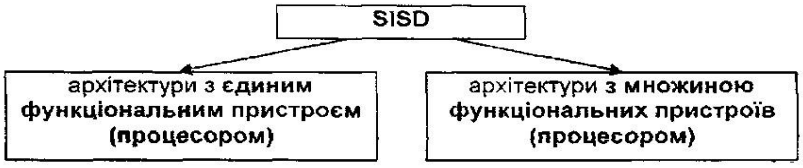


Рисунок 2.1 – Клас **SISD**

У клас **SIMD** уводяться два підкласи (рис.2.2):

- архітектури з **послівно-послідовною** обробкою інформації -ILLIAC IV, PEPE, BSP;
- архітектури з **розрядно-послідовною** обробкою - STARAN, ICL DAP.

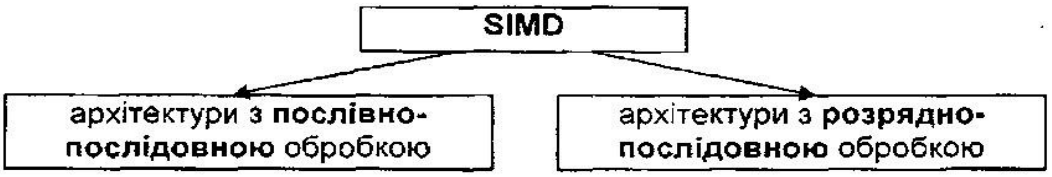


Рисунок 2.2 – Клас **SIMD**

У класі MIMD розрізняють (рис.2.3):

- обчислювальні системи зі **слабким зв'язком між процесорами (з розподіленою пам'яттю),** наприклад, Cosmic Cube,
- обчислювальні системи **із сильним зв'язком (з загальною пам'яттю)** C.mmp, BBN Butterfly, CRAY Y-MP, Denelcor HEP **У класі MIMD** розрізняють
- обчислювальні системи зі **слабким зв'язком між процесорами (з розподіленою пам'яттю),** наприклад, Cosmic Cube,
- обчислювальні системи **із сильним зв'язком (з загальною пам'яттю)** C.mmp, BBN Butterfly, CRAY Y-MP, Denelcor HEP

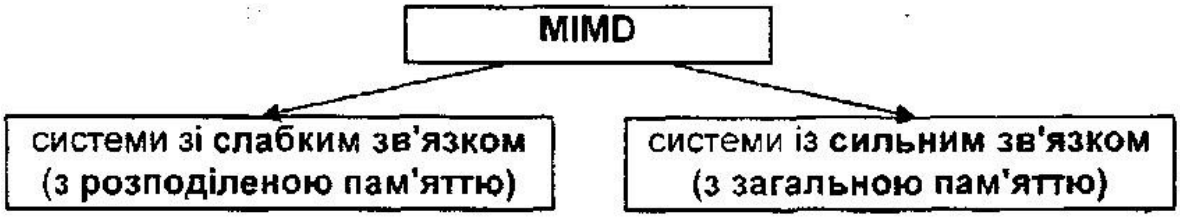


Рисунок 2.3 – Клас **MIMD**

2.3 Класифікація Хокни (доповнення МКМД Флінна)

Множинний потік команд може бути оброблений двома способами(рис.2.4):

- одним конвеєрним пристроєм обробки, що працює в режимі поділу часу для окремих потоків;
- кожен потік обробляється окремо виділеним пристроєм.

Перша можливість використовується в MIMD комп'ютерах, визначених конвеєрними (наприклад, процесорні модулі в Denelcor HEP). Архітектури, що використовують другу можливість, поділяються на два класи:

- MIMD комп'ютери, у яких;можливі прямі зв'язки кожного процесора з кожним, що реалізовані за допомогою комутатора;
- MIMD комп'ютери, у яких прямий зв'язок кожного процесора можливий тільки з найближчими сусідами по мережі, а взаємодія віддалених процесорів підтримується спеціальною системою маршрутизації через процесоріпосередники.

Серед MIMD-систем з комутатором виділяються ті, у яких уся пам'ять розподілена поміж процесорами як локальне (наприклад, PASM, PRINGLE). У цьому випадку взаємодія самих процесорів здійснюється за допомогою дуже складної системи комутування, що складає значну частину комп'ютера. Такі машини визначаються як MIMD-системи з розподіленою пам'яттю.

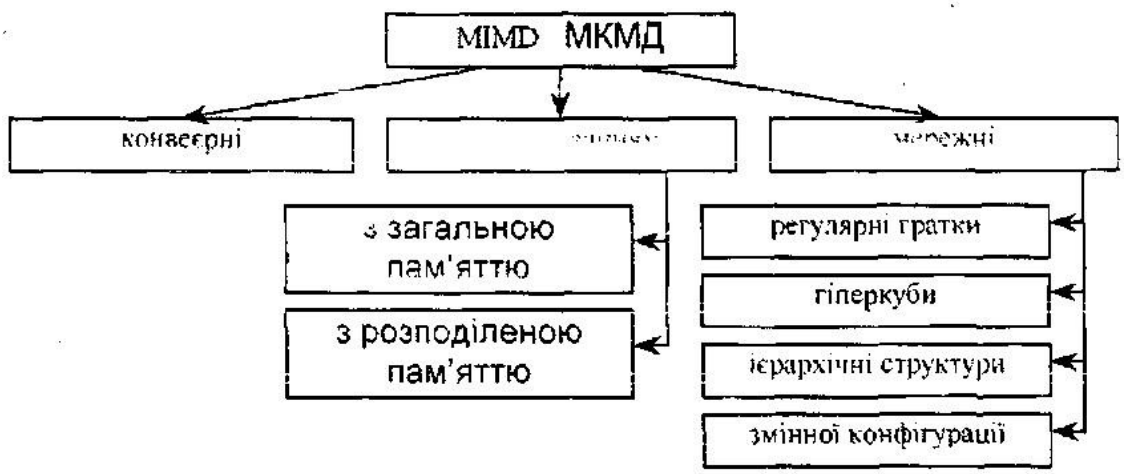


Рисунок 2.4 – Клас MIMD МКМД

Системи, що використовують як загальну поділювану пам'ять, так і розподілену локальну, визначаються як гібридні MIMD з комутуванням.

При аналізі MIMD-систем з мережною структурою вважається, що усі вони мають розподілену пам'ять, а подальша класифікація здійснюється відповідно до топології мережі:

- зіркоподібна мережа (ICAP);
- регулярні ґратки різної розмірності (Intel Paragon, CRAY T3D);
- гіперкуби (NCube, Intel iPCS);
- ієрархічні мережі типу дерева, піраміди, кластера (Cm*, CEDAR); мережі, зі змінною конфігурацією.

Якщо архітектура комп'ютера спроектована з використанням декількох мереж з різною топологією, то їх назвають гібридними мережними MIMD, а при використанні різних методів в системі - просто гібридними MIMD. Наприклад, комп'ютер Connection Machine 2 має на зовнішньому рівні топологію гіперкуба, кожен вузол якого є кластером процесорів з повним зв'язком.

2.4 Класифікація Джонсона

Класифікація MIMD-архітектур на основі аналізу структури пам'яті та реалізації механізму взаємодії і синхронізації між процесорами.

За структурою оперативної пам'яті ОС поділяються на дві групи:

- системи з загальною пам'яттю, яка прямо адресується усіма процесорами;
- системи з розподіленою пам'яттю, визначена частина якої доступна тільки одтному процесору.

Одночасно для міжпроцесорної взаємодії існують два методи:

- через поділювані змінні;
- за допомогою механізму передачі повідомлень.

Виходячи з вказаних припущень, одержуються чотири класи MIMD-архіектур, що уточнюють систематику Флінна:

- загальна пам'ять - поділювані змінні (GMSV);
- розподілена пам'ять - поділювані змінні (SMSV);
- розподілена пам'ять - передача повідомлень (SMMP);
- загальна пам'ять - передача повідомлень (GMMP).

Згідно наведеного розподілу, вводяться наступні визначення класів ОС.

• клас 1 - ОС, що використовують загальну поділювану пам'ять для міжпроцесорної взаємодії і синхронізації, визначаються системами з поділюваною пам'яттю, наприклад, CRAY Y-MP.

• клас 2 - системи, у яких пам'ять розподілена по процесорах, а для взаємодії і синхронізації використовується механізм передачі повідомлень, визначаються архітектурами з передачею повідомлень, наприклад NCube.

• клас 3 - системи з розподіленою пам'яттю і синхронізацією через поділювані змінні, як у BBN Butterfly, називаються гібридними архітектурами.

У класифікації визначена можливість врахування виду зв'язку між процесорами:

• загальна шина,
• системи комутування,
• різного роду мережі.

2.5 Класифікація Фенга

Класифікація Т.Фенга (1972 р.) здійснюється на основі двох характеристик (рис.2.5):

- число біт *n* у машинному слові, що обробляються паралельно при виконанні машинних інструкцій (у сучасних комп'ютерах це число збігається з довжиною машинного слова);числу слів *m*, що обробляються одночасно ОС.

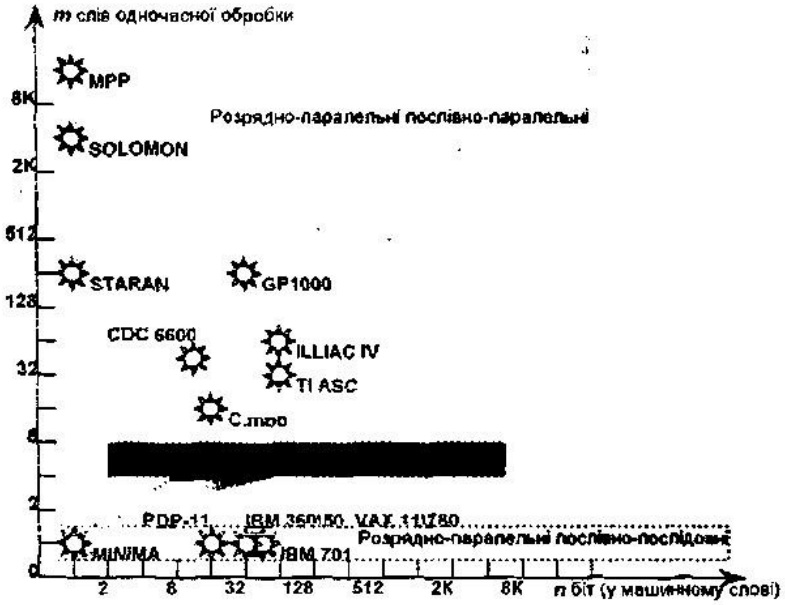


Рисунок 2.5 – Класифікація Т.Фенга

Функціонування ЗОТ можна подати як рівнобіжну (паралельну) обробку η бітових шарів, на кожному з який незалежно обробляються m біт. Друга характеристика визначає ширину бітового шару.

У координатній прив'язці до граничних верхніх значень даних характеристик кожну обчислювальну систему С можна подати парою чисел (n, m) і представити крапкою на площині в системі координат "довжина слова - ширина бітового шару".

Площа прямокутника зі сторонами *n* і *m* визначає інтегральну характеристику потенціалу паралельності Р архітектури і зветься максимальним ступенем паралелізму ОС: P(C)= nm, яка є піковою продуктивністю, вираженою в інших одиницях.

Наприклад, комп'ютер Advanced Scientific Computer фірми Texas Instruments (TI ASC) в основному режимі працює паралельно з 64-х розрядними словами. АЛП має чотири

одночасно працюючі конвеєри, що містять по вісім ступіней, що дає 4x8=32 біти в кожнім бітовому шарі. Тому комп'ютер TI ASC може бути представлений у виді (64,32).

На основі введених понять всі ОС в залежності від способу обробки, закладеного в архітектуру, можна розділити на чотири класи.

• Розрядно-послідовні послівно-послідовні (n - m =1). У кожен момент часу обробляється тільки один двійковий розряд. Система MINIMA описується як (1,1).

• Розрядно-паралельні послівно-послідовні (n > 1, т = 1). Більшість класичних послідовних комп'ютерів: IBM 701 = (36,1); PDP-11 = (16,1); IBM 360/50 і VAX 11/7SG - обидві = (32,1).

• Множина однорозрядних процесорних елементів, кожний з який незалежно від інших обробляє свій потік даних. Типові ОС: STARAN = (1, 256) і MPP =

(1,16384) фірми Goodyear Aerospace; прототип ОС ILLIAC IV -SOLOMON = (1, 1024) і ICL DAP = (1, 4096).

• Значна частина існуючих рівнобіжних ОС, що обробляють одночасно nt двійкових розрядів. ILLIAC IV = (64, 64); TI ASC = (64, 32); C.mmp = (16, 16); CDC 6600 = (60, 10); BBN Butterfly GP1000 = (32, 256).

Недоліки класифікації зв'язані зі способом обчислення ширини бітового шару т. Не відображено ніякого розходження між процесорними матрицями, векторно-конвесрними і багатопроцесорними системами, а також за рахунок чого може одночасно обробляти більше одного слова: множинності функціональних пристроїв, їх конвеєрності чи ж якогось числа незалежних процесорів. Якщо в системі N незалежних процесорів мають кожний по F конвеєрних функціональних пристроїв з довжиною конвеєра L, то для обчислення ширини бітового шару треба просто знайти добуток даних характеристик.

Досить важко (а іноді і неможливо) усвідомити специфіку організації певної ОС є введення єдиної числової метрики для всіх типів комп'ютерів, що разом з описом потенціалу обчислювальних можливостей конкретної архітектури дозволяє порівняти будь-які два комп'ютери між собою.

2.6 Класифікація Шора (розширення Фенга)

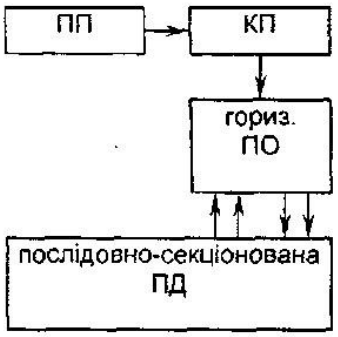
Класифікація Дж. Шора (початок 70-х р.) виділяє типові способи компонування ОС на основі фіксованого числа базисних блоків:

- пристрою керування КП;
- процесора ПО;
- пам'яті команд ПК;
- пам'яті даних ПД.

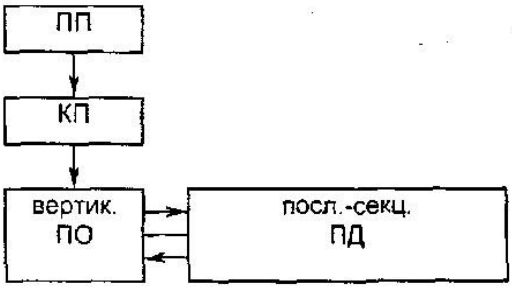
Також передбачається, що вибірка з пам'яті даних може здійснюватися словами, (вибираються всі розряди одного слова), і/чи бітовим шаром (по одному розряду з однієї і тієї ж позиції кожного слова). Іноді ці два способи називають горизонтальною і вертикальною вибірками відповідно. Відповідно до критеріїв розрізняють шість класів ЗОТ:

Машина I (Рис.2.6), (послівно-послідовна, розряднопаралельна) - це ОС з послідовною вибіркою всіх розрядів слів із єдиної ПД і паралельною обробкою їх розрядів в процесорі ПО Склад ПО не обумовлюється, що допускає наявність кількох функціональних пристроїв (наприклад, конвеєрного типу). У даний клас попадають як класичні послідовні машини

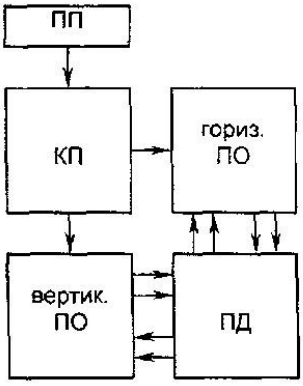
(IBM 701, PDP-11, VAX 11/780), так і конвеєрні скалярні (CDC 7600) і векторно-конвеєрні (CRAY-1).



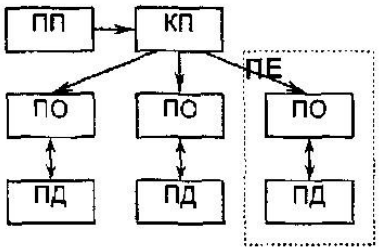
Машина II (Рис.2.7), (послівно-паралельна, розрядно-послідовна)-ОС із вибіркою не за словами, а послідовно за вмістом одиничних розрядів та їх паралельною обробкою для усіх слів. Слова в ПД розташовані горизонтально, але реалізована послідовна обробка бітових шарів при паралельній рівнобіжній обробці безлічі слів.



Зчитування даних із ПД здійснюється із розрядної секції всіх слів пам'яті, а не всіх розрядів одного слова. Процесор організований для виконання операцій розрядно-послідовним методом. Прикладом є асоціативні комп'ютери (центральний процесор ОС STARAN). Такі комп'ютери мають не один ПО, а безліч порівняно простих пристроїв порозрядної обробки. Матрична ОС ICL DAP потенційно здійснює порозрядну обробку до 4096 слів.

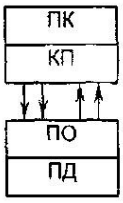


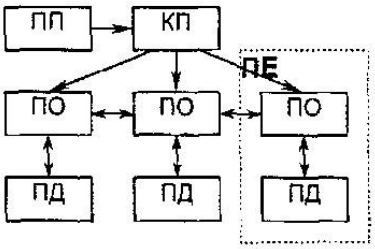
Машина III (Рис.2.8), (ортогональний комп'ютер Шумана 1960 р.) - ОС з узагальненими принципами побудови машин I і II. Модифікована двовимірна ПД забезпечує доступ як до слів, так і до бітових шарів (розрядними секціями). Якщо пам'ять подати як матрицю слів, то доступ до даних здійснюється в напрямку, "ортогональному традиційному - не тільки словами (рядкам), а бітовими шарами (стовпцями). Із двох наявних ПО із горизонтальною організацією обробляє слова, а ПО із вертикальною - розрядні секції. Прикладом є ОС сімейства OMEN-60 фірми Sanders Associates.



Машина IV (Рис.2.9), (ОС на основі незв'язаної матриці НзМ) є системою процесорних елементів ПЕ, які складаються власне з процесора ПО та пам'яті даних ПД, що функціонують під керуванням єдиного пристрою керування КП згідно потоку інструкцій ПК, зчитаних з пам'яті програм ПП. ПЕ в ОС відносно просто нарощуються до необхідної розмірності внаслідок відсутності з'єднань між ПЕ. Прикладом є ОС PEPE (288 ПЕ).

Машина V (Рис.2.10), (ОС на основі зв'язаної матриці ЗвМ) -є сукупністю взаємозв'язаних ПЕ, довільний із яких може звертатись до вмісту як своєї пам'яті, так і пам'яті сусідніх ПЕ, всі із них знаходяться під керуванням єдиного пристрою керування із пам'яті програм. Прикладом є матричний комп'ютер ILLIAC IV.





Машина VI (Рис.2.11), (матрична ОС із функціональною пам'яттю - вбудованою логікою) Використано метод розподілу функціональної логіки процесора по всьому запам'ятовуючому пристрою. Прикладами є як прості асоціативні запам'ятовуючі пристрої, так і складні асоціативні процесори.

2.7 Класифікація Хендлера (розширення Ерланген)

В основу закладено критерії рівнобіжної і конвеєрної обробки інформації. Упереджено не розглядаються можливі методи зв'язку між процесорами і блоками пам'яті. Вважається, що

комунікаційна мережа може бути сконфігурована потрібним чином і в стані витримати передбачуване навантаження.

Класифікація базується на специфіці трьох рівнів обробки даних у процесі виконання програм:

- ***рівень виконання програми*** - контролюючи лічильник команд і деякі інші регістри, пристрій керування КП здійснює вибірку і дешифрування команд програми;
- ***рівень АЛП***- арифметико-логічний пристрій АЛП комп'ютера виконує команду, задану йому пристроєм керування КП;
- ***рівень бітової обробки*** - всі елементарні логічні схеми ЕЛС процесора розбиваються на групи, необхідні для виконання операцій над одиничними двійковими розрядами.

Допускається, що ОС включає в конвеєрі певне число процесорів Ppk, кожний з автономним КП, які зв'язані з кількома АЛПd, що виконують ту саму операцію в кожен поточний момент часу. Кожен з АЛП поєднує кілька ЕЛС, асоційованих з обробкою одиничного двійкового розряду (число ЕЛС = довжині машинного слова).

Не аналізуючи типу конвеєризації, кожну ОС можна подати трійкою характеристичних параметрів (Ерлангерський триплет):

- k - число процесорів (із пристроями керування КП) в конвеєрі, що одночасно інтерпретують програму;
- d - число АЛП, керованих одним із k пристроїв керування КП;
- w - розрядність слова, або число ЕЛС у кожному з d АЛП:

$$t(C) = (k, d, w)$$

Незважаючи на те, що в системах закладений паралелізм різного роду, він може бути легко віднесений до одного з трьох виділених рівнів.

Наведена Ерланген-характеристика дозволяє трактувати параметри довільної ОС у тривимірному просторі координат як прямокутну фігуру із розмірами сторін, пропорційними значенням параметрів, об'єм якої визначає потенційний ступінь паралелізму в термінах Фенга, або відносну пікову продуктивність (рис.2.12).

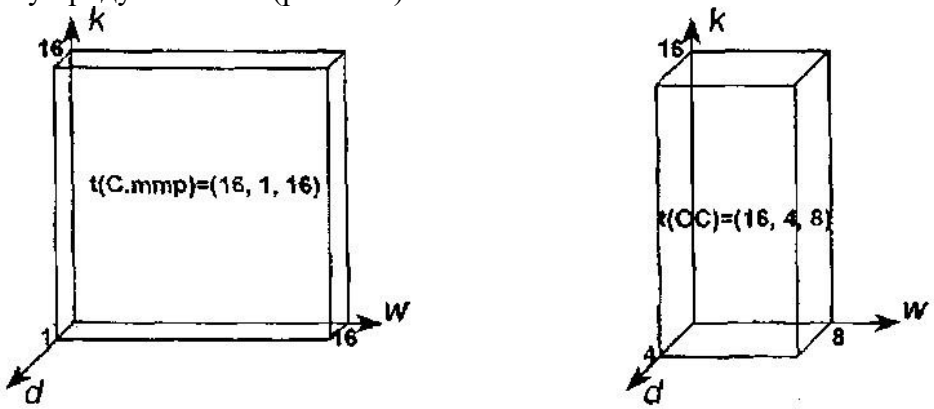


Рисунок 2.12 – Трактування Ерлангерського триплету у тривимірному просторі координат

Розширимо можливості опису, передбачивши можливість конвеєрної обробки на кожному з рівнів.

Конвеєризація на самому нижньому рівні (тобто на рівні ЕЛС) це конвеєризація функціональних пристроїв. Якщо *w'* функціональних пристроїв ФП обробляють w-розрядні слова на кожному з w' ступіней конвеєра, то для характеристики паралелізму даного рівня природно розглянути добуток *wxw'*. Знак множення χ використовується на кожному рівні з метою відокремлення числа, що визначає склад обчислювального ресурсу від числа ступіней у конвеєрі.

Комп'ютер TI ASC має чотири конвеєрних пристрої по вісім ступіней у кожному для обробки 64-х розрядних слів, тому він описується як: *t(TIASC) = (1,4, 64x8)*

Наступний рівень конвеєрної обробки - **це *конвеєризація* .**Передбачається, що в ОС є кілька ФП, які можуть працювати одночасно в рамках одного потоку команд (зачеплення ФП). Історично першою була система CDC 6600, що містить 10 незалежних послідовних ФП, утворивши в конвеєрі єдиний потік команд: *t(CDC 6600) = (1, 1x10, 64)*

(центральний процесор без керуючих і периферійних підсистем).

Конвеєризація на самому верхньому рівні, інакше макро-конвейер. Потік даних, проходячи через один процесор ЦП, надходить на вхід іншого, можливо через деяку буферну пам'ять. Якщо незалежно працюють η процесорів, то в ідеальній ситуації при відсутності конфліктів і повної збалансованості одержуємо прискорення в *η* разів у порівнянні з використанням тільки одного процесора. Комп'ютер PEPE, маючи фактично три незалежних процесорні системи з 288-ми пристроїв обробки, описується як

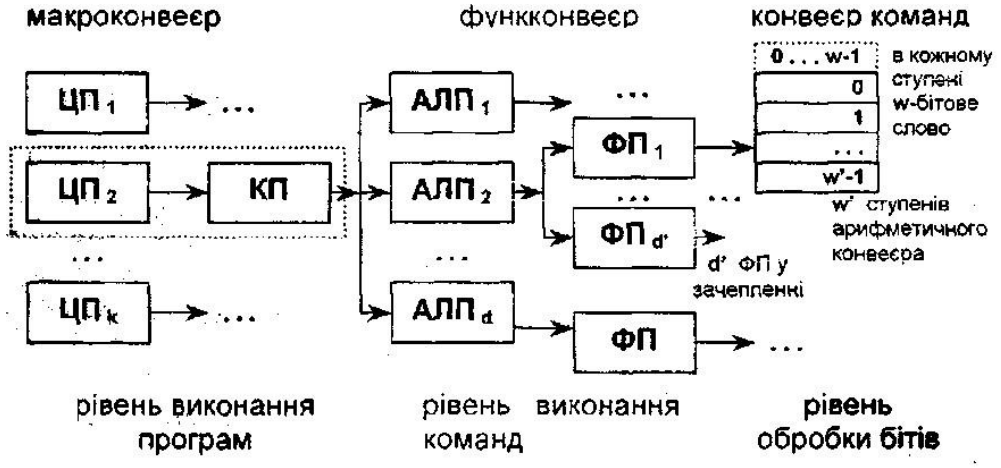
t(PEPE) = (1x3, 288, 32)

Після розширення трирівневої моделі паралелізму (рис.2.13) засобами опису потенційних можливостей конвеєризації кожна трійка

t(OC) = (kxk', dxd', wxw')

інтерпретується так:

k - число процесорів, що працюють в системі конвеєра; *k'* — число конвеєрних систем в макроконвеєрі (довжина макроконвеєра); *d*- число АЛП під керуванням одного із КП в кожному процесорі; *d'*-число функціональних пристроїв в складі одного АЛП, що обробляють один потік даних (довжина конвеєра команд); w- чисіпо' розрядів у слові, що обробляються в АЛП паралельно; *w'*-число ступіней арифметичного конвеєра виконання однієї команди (довжина конвеєра мікрокоманд).



t(Intel Paragon XP/S) = (1840×2, 1×2, 64×3)

Рисунок 2.13 – Розширення трирівневої моделі паралелізму.

Зв'язок між класифікацією Фенга і класифікацією Хендлера: Максимального ступінь паралелізму в термінах Фенга визначається добутком шести величин в описі Хендлера. Явна вказівка на присутній паралелізм і можливу конвеєризацію знімає питання, характерні для схем Флінна, Шора і Фенга, у плані опису векторно-конвеєрних машин.

Запропоновані три формальні операції, дозволяють також описати:

• складні структури з процесорними підсистемами вводу-виведення, хосткомпьютером чи іншими особливостями;

• можливі режими функціонування ОС, що підтримуються для оптимальної відповідності щодо структури виконуваних програм.

Перша операція (x) характеризує конвеєрний принцип обробки і припускає послідовне проходження даних спочатку через перший аргумент-підсистему, а потім через другий і т.д.

Згаданий комп'ютер CDC 6600 можна уточнити як:

t(*CDC 6600*) = (*10, 1, 12*) *x*(*1, 1x10, 64*),

де перший аргумент трактує десять 12-ти розрядних периферійних процесорів *i*, що кожна програма повинна спочатку бути оброблена одним *з* них і лише після цього передана ЦП для виконання. Аналогічно для машини PEPE, в комплексі хост-компьютера CDC 7600:

t(PEPE)=t(*CDC* 7600X7x3, 288, 32)=(15, 1, 12)*x*(1, 1*x*9, 60)*x*(1*x*3, 288, 32)

Всі підсистеми останнього прикладу досить складні і, виходячи тільки з даного опису можуть бути представлені по-різному. Аналогічно операції конвеєрного виконання, Друга операція рівнобіжного виконання (+), фіксує можливість незалежного використання процесорів різними задачами:

$$t(n,d,w) = [\{(1,d,w) + ... + (1,d,w)\}] \{n \text{ раз}\}.$$

У випадку CDC 7600 уточнений запис набуває виду:

(15, 1, 12)*x*(1, 1*x*9, 60)=[(1, 1, 12)+... +(1, 1, 12)]} {15 *разів*} *x*(1, 1*x*9, 60)

говорить про те, що кожна задача може захопити свій периферійний процесор, а потім вони одна за однією вони будуть надходити в ЦП.

Третя операція - операція альтернативи (V), показує можливі альтернативні режими функціонування обчислювальної системи. Чим більше для системи властиво таких режимів, тим більше гнучкою архітектурою вона володіє. Наприклад, комп'ютер C.mmp може бути запрограмований для використання в трьох принципово різних режимах: *t*(*C.mmp*) = (16, 1, 16) *V*(1*16, 1, 16) *V*(1, 16, 16).

2.8 Класифікація Базу

А.Базу (A.Basu) Будь-яку рівнобіжну ОС можна однозначно описати послідовністю рішень, прийнятих на етапі її проектування, а сам процес проектування представити у виді дерева, в якому корінь - це ОС, а наступні яруси дерева, фіксуючи рівень паралелізму, метод реалізації алгоритму, паралелізм інструкцій і спосіб керування, послідовно доповнюють один одного, формуючи специфікацію системи (рис.2.14).

На **першому етапі** визначається, який рівень паралелізму використовуватиметься в ОС. Та сама операція може одночасно виконуватися над цілим набором даних, визначаючи паралелізм на рівні даних (позначається буквою D на малюнку). Здатність виконувати більше однієї операції одночасно говорить про паралелізм на рівні команд (буква O на малюнку). Якщо ж комп'ютер спроектований так, що цілі послідовності команд можуть бути виконані одночасно, то будемо говорити про паралелізм на рівні задач (буква T).

Другий рівень у класифікаційному дереві визначає метод реалізації алгоритму. З появою надвеликих інтегральних схем (НВІС) стало можливим реалізовувати апаратно не тільки прості арифметичні операції, але й цілком алгоритми. Наприклад, швидке перетворення Фур'є, добуток матриць і LU-розкладання відносяться до класу тих алгоритмів, що можуть бути ефективно реалізовані на НВІС'ах. Даний рівень класифікації розділяє системи з апаратною реалізацією алгоритмів (буква C на схемі) і системи, що використовують традиційний спосіб програмної реалізації (буква P).

Третій рівень конкретизує тип паралелізму, використаного для обробки команд машини: конвеєризація команд (Pi) чи їх незалежне (рівнобіжне) виконання (РЙ), У більшій мірі цей вибір відноситься до комп'ютерів із програмною реалізацією алгоритмів, тому що апаратна реалізація завжди припускає рівнобіжне виконання команд. Відзначимо, що у випадку конвеєрного виконання мається на увазі лише конвеєризація самих команд, що розбиває весь цикл обробки на вибірку команди, дешифрування, обчислення адрес і т.д., - можлива конвеєризація обчислень на даному рівні не приймається до уваги. Четвертий рівень

визначає спосіб керування, прийнятий в ОС: синхронний (S) чи асинхронний (A). Якщо виконання команд відбувається в строгому порядку, обумовленому тільки сигналами таймера і лічильником команд, то будемо говорити про синхронний спосіб керування. Якщо ж для ініціації команди визначальними є такі фактори, як, наприклад, готовність даних, то попадаємо в клас машин з асинхронним керуванням. Найбільш характерними представниками систем з асинхронним керуванням є data-driven і demand-driven комп'ютери.

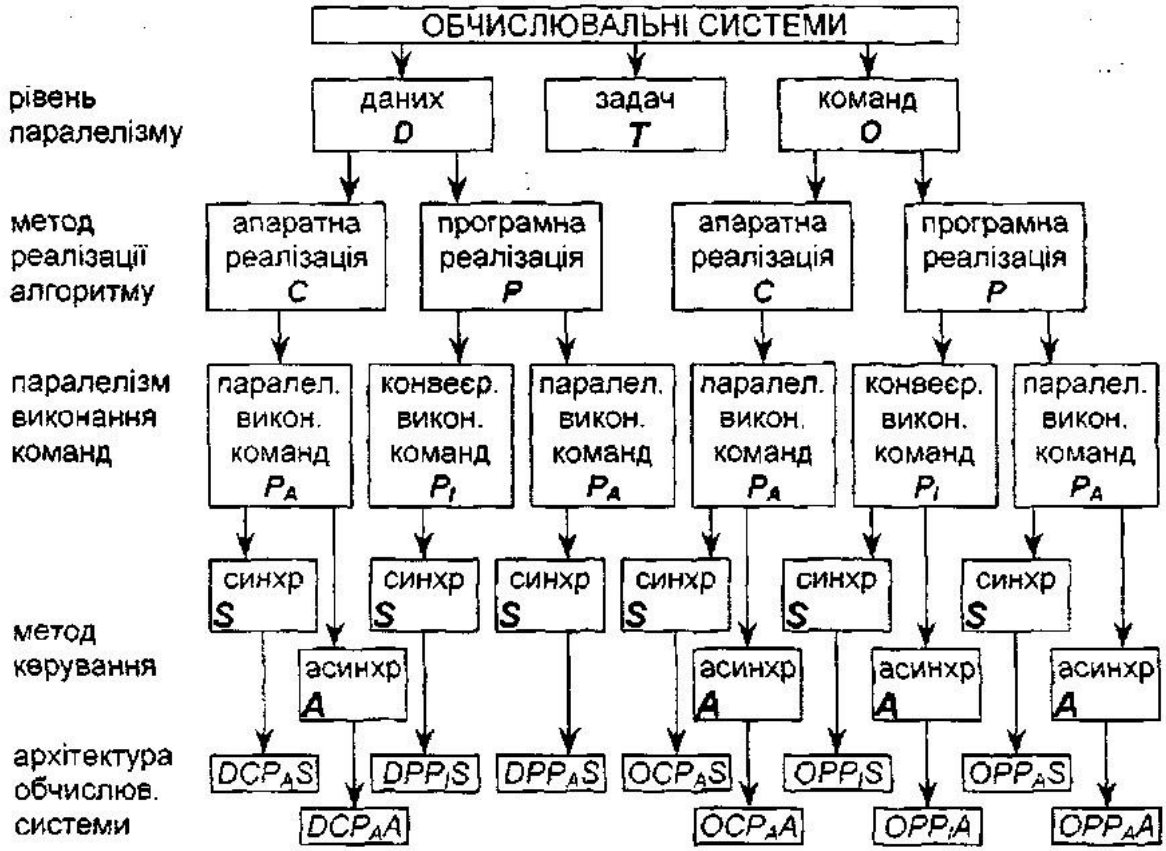


Рисунок 2.14 – Класифікація Базу

Проаналізувавши основні принципи класифікації, визначимо, куди попадають різні типи рівнобіжних ОС.

Вивчення систолічних масивів, що мають, як правило, одномірну чи двовимірну структуру, показує, що позначення DCPaS і DCPaA можуть бути використані для їхнього опису в залежності від того, як відбувається обмін даними: синхронно чи асинхронне. Систолічні дерева, введені Кунгом для обчислення арифметичних виразів, можуть бути описані як OCPaS або OCPaA згідно аналогічних міркувань. Конвеєрні комп'ютери, такі, як IBM 360/91, Amdahl 470/6 і багато сучасних RISC-процесорів, що розбивають виконання всіх інструкцій на кілька етапів, у даній класифікації мають позначення OPP/S. Більш природне застосування конвеєризації відбувається у векторних машинах, у яких одна команда застосовується до вектора незалежних даних, і за рахунок безупинного використання арифметичного конвеєра досягається значне прискорення.

До таких комп'ютерів підходить позначення DPPiS. Матричні процесори, у яких множина арифметичних пристроїв працює одночасно в строго синхронному режимі, належать до групи DPPaS. Якщо ОС типу CDC 6600 має процесор з окремими функціональними пристроями, що керуються централізовано, то її опис виглядає так: OPPaS. Data-flow

комп'ютери, у залежності від особливостей реалізації, можуть бути описані як OPPiAi або OPPaA·

Системи з декількома процесорами, що використовують паралелізм на рівні задач, не завжди можна коректно описати в рамках запропонованого формалізму. Якщо процесори додатково не використовують паралелізм на рівні операцій чи даних, то для опису можна використовувати лише букву T, В іншому випадку Базу пропонує використовувати знак '*' між символами, що позначають рівні паралелізму, що одночасно використовуються системою. Наприклад, комбінація T*D означає, що деяка система може одночасно виконувати кілька задач, причому кожна з них може використовувати векторні команди.

Дуже часто в реальних системах присутні особливості, характерні для комп'ютерів з різних груп даної класифікації. У цьому випадку для коректного опису автор використовує знак '+'. Наприклад, практично усі векторні комп'ютери мають скалярну і векторну частини, що можна описати як OPPiS+DPPiS (приклад - це TI ASC і CDC STAR-100). Якщо в системі є можливість одночасного виконання більш однієї векторної команди (як у CRAY-1) то для опису векторної частини можна використовувати запис O*DPPiS, а повний опис даного комп'ютера виглядає так: O*DPPiS+OPPiS. Діючи за таким принципом, можна визначити і системи CRAY X-MP і CRAY Y-MP, які поєднують кілька процесорів, що мають схожу з CRAY-1 структуру, тому їхній опис має вид: T*(O*DPPiS + OPPiS).

2.9 Класифікація Шнайдера

Класифікація Л.Шнайдера (L. Snyder) (1988 р.) опису архітектур рівнобіжних обчислювальних систем, що попадають у клас SIMD систематики Флинна, полягає у виділенні етапів вибірки і безпосередньо виконання в потоках команд і даних. Поділ потоків на адреси і їхній вміст дозволяє описати такі раніше "незручні" для класифікації архітектури, як комп'ютери з довгим командним словом, систолічні масиви і ряд інших.

Визначимо потоком посилань (reference stream) S деякої ОС кінцеву множину нескінченних послідовностей пар:

$S = \{(a_1 <t_1>)(a_2<t_2>)..., (b_1 <u_1>)(b_2<u_2 >)..., (c_1<v_1>)(c_2<v_2 >)...\}$,

де перший компонент кожної пари - ціле число, назване адресою, другий компонент - це набір з n додатних цілих чисел, названих значеннями, причому n однакове для всіх наборів усіх послідовностей. Наприклад, пара (b2<и2>) визначає адресу b2 і значення <и2>. Якщо значення розглядати як команди, то з потоку посилань одержимо потік команд і; якщо ж значення інтерпретувати як дані, то відповідний потік - це *потік даних* D.

Інтерпретація введених понять полягає у наступному. Елементи кожної послідовності - адреса і її вміст, зчитані з (чи записувані в) пам'ять. Послідовність пар адреса-значення можна розглядати як історію виконання команд або переміщення даних між процесором і пам'яттю комп'ютера під час виконання програми. Число інструкцій, що даний комп'ютер може виконувати одночасно, визначає число послідовностей у потоці команд. Аналогічно, число різних даних, що комп'ютер може обробити одночасно, визначає число послідовностей у потоці даних. Нехай S - довільний потік посилань. Послідовність адрес потоку S що позначається Sa, - це послідовність, чий і-й елемент - набір, сформований з адрес і-х елементів кожної послідовності з S:

Sa = <a1 b1 ...c1 >, <a2 b2 ...c2 > ,...

потоку S, означена Sv - послідовність, чий і-й елемент - набір, утворений злиттям наборів значень і-х елементів кожної послідовності з S:

Sv= <t1 u1 ...v1 >, <t2 u2 ...v2 > ,...

Якщо Sx - послідовність, елементів, де кожен елемент - набір з η чисел, "ширини" послідовності позначається як: w(Sx) = n.

З визначень Sa, Sv і w випливає твердження:

якщо S - це потік посилань зі значеннями з η чисел, то

$$w(Sa) = |S|\ w(Sv) = n|S|,$$

де |S| позначає потужність множини S.

Кожну пару (*I*, D) з потоком команд I потоком даних D будемо називати обчислювальним шаблоном, а всі комп'ютери будемо розбивати на класи в Залежності від того, який шаблон вони можуть виконати. Справді, комп'ютер може виконати шаблон (I, D), якщо він у стані:

- видати $w(I_a)$ адрес команд для одночасної вибірки з пам'яті;
- декодувати і проінтерпретувати одночасно w(lv) команд;
- видати одночасно $w(D_a)$ адрес операндів;
- виконати одночасно $w(D_v)$ операцій над різними даними.

Якщо всі умови виконані, то комп'ютер описується у такий спосіб:

Iw(ta)w(lv)Dw(Da)w(Dv)

Розглянемо класичну послідовну машину. Відповідно до класифікації Флінна, вона попадає в клас SISD, отже $|I| = |D| = 1$. Використовуючи твердження 1, одержуємо, що $w(I_a) = w(D_a) = 1$. Через те, що в подібного роду комп'ютерах команди декодуются послідовно, випливає рівність $w(I_v) = 1$, а послідовне виконання команд дає $w(D_v) = 1$. Тому опис однолроцесорної машини з фон-неймановською архітектурою буде виглядати так:

I1,1D1,1

При аналізі ОС із класу SIMD: Goodyear Aerospace MPP і ILLIAC IV не береться до уваги різниця в способах обробки даних окремими процесорними елементами. Єдиний потік команд означає $|I| = 1$ для обох ОС. Для потоку команд одержуємо рівність $w(I_a) = w(I_v) = 1$. Далі, для доступу до операндів пристрій керування MPP розсилає ту саму адресу всім процесорним елементам, тому в цій термінології MPP має єдину послідовність у потоці даних, тобто $|D| = 1$. Однак потім вибірка даних з пам'яті і наступна обробка здійснюється в кожному процесорному елементі, тому $w(D_v)$=16384, а вся система MPP описується як:

I1,1D1,16384

В ILL1AC IV пристрій керування, так само, як і в MPP, розсилає ту саму адресу всім процесорним елементам, однак кожний з них може одержати свою унікальну адресу, додаючи вміст локального індексного регістра. Це означає, що $|D| = 64$ і в системі присутні

64 потоки адрес даних, що визначають одиночні потоки операндів, тобто $w(D_a) = w(D_v) = 64$.

Підсумовуючи сказане, приходимо до опису ILLIAC IV:

I1,1D64,64

Для більш чіткої класифікації уводиться три предикати для позначення значень, що можуть приймати величини $w(I_a)$, $w(I_v)$, $w(D_a)$ і $w(D_v)$:

- s - предикат "дорівнює 1";
- c - предикат "від 1 до якоїсь (невеликої) константи";
- *m* - поедикат "від 1 до довільно великого кінцевого числа".

У цих позначеннях, наприклад, фон-неймановська машина належить до класу IssDss. Незважаючи на те, що і 'c' і 'т' у принципі не мають визначеної верхньої границі, вони відбивають різні властивості архітектури комп'ютера. Визначник 'c' припускає тверді обмеження зверху з боку апаратури, і відповідний параметр не може бути значно збільшений відносно простими засобами. Прикладом може служити число інструкцій, упакованих у командному слові VLI комп'ютера. З іншого боку, визначник 'т' використовується тоді, коли величина, що позначається, може бути легко змінена, тобто, комп'ютер за даним параметром масштабується. Наприклад, відносна простота в збільшенні числа процесорних елементів у системі MPP є підставою для того, щоб віднести її до" класу lssDsm. Звичайно ж, розходження між 'c' і 'т' у достатній мірі умовне і; як правило, породжує масу питань. Зокрема, як описати машину, у якій процесори зв'язані через загальну шину? З одного боку; немає ніяких

принципових обмежень на число процесорів, що підключаються. Однак кожен додатковий процесор збільшує завантаженість шини, і при досягненні деякого порога, підключення нових процесорів немає сенсу. Як описати таку систему, 'c', чи 'm' - дане питання залишається відкритим.

На основі зазначених предикатів можна виділити наступні класи комп'ютерів:

$I_{ss}D_{ss}$ - фон-неймановські машини;

$I_{ss}D_{sc}$ - фон-неймановські машини, у яких закладена можливість вибирати дані, розташовані з різним зсувом щодо тієї самої адреси, над якими буде виконана та сама операція. Прикладом можуть служити комп'ютери, що мають команди, типу одночасного виконання двох операцій додавання над даними у форматі півслова, розташованими за зазначеною адресою.

$I_{ss}D_{sm}$ - SIMD-комп'ютери без можливості одержання унікальної адреси для даних у кожному процесорному елементі, що включають MPP, Connection Machine 1 так само, як і систолічні масиви.

$I_{ss}D_{cc}$ - багатовимірні SIMD-машини - фон-неймановські машини, здатні розщеплювати потік даних на незалежні потоки операндів;

$I_{ss}D_{mm}$ - це SIMD-комп'ютери, що мають можливість незалежної модифікації адрес операндів у кожному процесорному елементі, наприклад, ILLIAC IV і Connection Machine 2.

$I_{sc}D_{cc}$ - обчислювальні системи, що вибирають і виконують одночасно кілька команд, для доступу до яких використовується одна адреса. Типовим прикладом є комп'ютери з довгим командним словом (VLI).

$I_{cc}D_{cc}$ - багатовимірні MIMD-машини. Фон-неймановські машини, що можуть розщеплювати свій цикл вибірки/виконання з метою обробки паралельно декількох незалежних команд.

$I_{mm}D_{mm}$ - до цього клзсу відносяться всі комп'ютери типу MIMD.

Досить ясно, що не потрібно розглядати всі можливі комбінації визначників 's', 'c' і 'т', тому що архітектура реальних комп'ютерів накладає ряд цілком розумних обмежень. Очевидно, що число адрес w(Sa) не повинне перевищувати числа повернутих значень w(Sv), що комп'ютер може обробити. Звідси випливають нерівності: $w(I_a) < w(I_v)$ і $w(D_a) < w(D_v)$. Іншим природним припущенням є той факт, що число виконуваних команд не повинне перевищувати числа оброблюваних даних: $w(I_v) < w(D_v)$.

Підводячи підсумок, можна відзначити два позитивних моменти в класифікації Шнайдера: більш вибіркова систематизація SIMD комп'ютерів і можливість опису нетрадиційних архітектур типу систолічних чи масивів комп'ютерів з довгим командним словом. Однак майже всі ОС типу MIMD знову потрапили в той самий клас /mmDmm , тому що критерій класифікації, який грунтується лише на потоках команд і даних без врахування розподілу пам'яті і топології міжпроцесорних зв'язків, виявляється недосконалий для подібних систем.

3. ПРЕДМЕТ І ЗАДАЧІ ТЕОРІЇ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

3.1. Предмет теорії

Теорія обчислювальних систем — інженерна дисципліна, що поєднує методи вирішення задач проектування й експлуатації ЕОМ, обчислювальних комплексів, систем і мереж.

Предмет теорії. Предметом дослідження в теорії обчислювальних систем є обчислювальні системи в аспектах їхньої продуктивності, надійності і вартості. У системі виділяються наступні складові:

1) технічні засоби, обумовлені конфігурацією системи — складом пристроїв і структурою зв'язків між ними;

2) режим обробки, що визначає порядок функціонування системи;

3) робоче навантаження, що характеризує клас оброблюваних задач і порядок їхнього надходження в систему.

Коли ЕОМ, обчислювальний комплекс, система чи мережа досліджується в цілому, як органічна єдність складових у взаємодії з навколишнім середовищем, і при цьому виявляються загальносистемні властивості і характеристики, говорять, що дослідження проводиться на системному рівні. Представлення досліджуваних об'єктів (ЕОМ, комплекси, системи і мережі) на системному рівні — найбільш характерна риса теорії обчислювальних систем.

Предметом дослідження може бути функціонування процесора, зовнішнього запам'ятовуючого пристрою і каналу введення — виведення, обмін даними між рівнями пам'яті, планування, обробка, системне введення — вивід і ін. При цьому властивості елементів і підсистем вивчаються стосовно до цілям дослідження всієї системи, наприклад до оцінки продуктивності, і розглядаються як частини системи, що функціонують у взаємодії з іншими частинами.

3.2. Задачі аналізу

Задачі аналізу. Аналіз обчислювальних систем — визначення властивостей, властивій чи системі класу систем. Типова задача аналізу — оцінка продуктивності і надійності систем із заданою конфігурацією, режимом функціонування і робочим навантаженням Інші приклади задач визначення (оцінка) імовірності конфлікту при доступі до загальної шини, розподілу тривалості зайнятості процесора, завантаження каналу введення — виведення

У загальному випадку задача аналізу формулюється в такий спосіб. Виходячи з мети дослідження призначається набір характеристик $Y=\{y_1, \ldots, y_{M1}\}$ досліджуваного об'єкта (обчислювальна система, її елемент, підсистема, деякий процес і ін.) і точність $\Delta = \{\delta_1, .., \delta_M\}$, з яким вони повинні бути визначені. Потрібно знайти спосіб оцінки характеристик Y об'єкта з заданою точністю Δ і на основі цього способу визначити характеристики.

При аналізі систем у процесі експлуатації оцінка характеристик Y здійснюється, як правило, виміром параметрів функціонування з обробкою вимірювальних даних. У цьому випадку використовується методика, що встановлює склад вимірюваних параметрів, періодичність і тривалість вимірів, а також вимірювальні засоби і засоби обробки даних З метою скорочення витрат на аналіз прагнуть вимірювати по можливості менше число найбільш просто вимірюваних параметрів $X=\{x_1,\ldots ,x_v\}$, а необхідний набір характеристик визначати непрямим методом — обчисленням з використанням залежностей $y_m= \varphi_m(X)$, $m=1, \ldots, M$. Ці залежності, або мають статистичну природу, або створюються на основі фундаментальних закономірностей теорії обчислювальних систем.

При аналізі проектованих систем для оцінки характеристик Y необхідно володіти моделлю F, що встановлює залежність $Y=F(X)$ характеристик від параметрів системи X, що

визначають її конфігурацію, режим функціонування, робоче навантаження. У цьому випадку розв’язок задачі зводиться до проведення на моделі експериментів, що дозволяють дати відповіді на цікавлячі питання. Точність оцінки характеристик проектованої системи залежить від адекватності моделі і похибки виміру параметрів *X*.

3.3. Задачі ідентифікації

Задачі ідентифікації. При експлуатації обчислювальних систем виникає необхідність у підвищенні їхньої ефективності шляхом підбору конфігурації і режиму функціонування, що відповідають класу розв'язуваних задач і вимогам до якості обслуговування користувачів. У зв'язку з зростанням навантаження на систему і переходом на нову технологію обробки даних може знадобитися зміна конфігурації системи, використовування більш сучасних операційних систем і реалізованих ними режимів обробки. У цих випадках варто оцінити можливий ефект, для чого необхідні моделі продуктивності і надійності системи. Побудова моделі системи на основі апріорних даних про її організацію до даних вимірів називається *ідентифікацією системи.*

Порядок ідентифікації обчислювальної системи ілюструється рис.3.1. Відповідно до природи досліджуваних явищ для їхнього представлення пропонується функціональна модель, що описує явища з точністю до значень параметрів функцій. Процес створення такої моделі називається *функціональною ідентифікацією* системи. Як функціональні моделі можуть використовуватися різні математичні системи — диференціальні й алгебраїчні рівняння, мережі масового обслуговування й ін., що адекватно представляють досліджувані аспекти.

Після того як обрана функціональна модель, необхідно визначити її параметри. Цей процес називається параметричною ідентифікацією. Для параметричної ідентифікації до обчислювальної системи підключаються необхідні вимірювальні засоби.

Одержувані дані використовуються системою оцінки параметрів і характеристик для обчислення параметрів X* і характеристик Y* системи, а також параметрів моделі $A=\{a_n\}$. Система оцінки являє собою набір програм для обробки вимірювальних даних, що реалізує набір методів оцінки параметрів і характеристик. Обчислені значення параметрів *A* вводяться в модель, цілком визначаючи неї. Значення параметрів X* і характеристик Y* системи використовуються для перевірки адекватності моделі, тобто оцінки похибки Δ відтворення моделлю характеристик системи. Оцінка здійснюється шляхом порівняння значень характеристик $Y=F(X^*)$, породжуваних моделлю, із зареєстрованими характеристиками Y* системи. Якщо модель адекватна системі, то використовується для прогнозування властивостей системи, що зводиться до обчислення на основі моделі характеристик $Y=F(X)$, що відповідають новим значенням *X* параметрів системи.

3.4. Задачі синтезу

Задачі синтезу. Синтез — процес створення обчислювальної системи, що щонайкраще відповідає своєму призначенню. Вихідними в задачі синтезу є наступні дані, що характеризують призначення системи: 1) функція системи (клас розв'язуваних задач); 2) обмеження на характеристики системи, наприклад на продуктивність, час відповіді, надійність і ін.; 3) критерій ефективності, що встановлює спосіб оцінки якості системи в цілому. Необхідно вибрати конфігурацію системи і режим обробки даних, що задовольняють заданим обмеженням і оптимальні за критерієм ефективності. Типова постановка задачі синтезу: спроектувати систему, що забезпечує розв’язок заданого класу задач ***A*** із продуктивністю не менш Λ задач на годину, середньою наробітком на відмовлення не менш $T0$ і мінімальною вартістю.

Математично задача синтезу обчислювальної системи формулюється в такий спосіб: $S = (s_1, \dots, s_p)$- вектор параметрів, що характеризують клас задач $\boldsymbol{A}$, розв'язок яких є функцією Нехай системи; S=(s_1,...,s_p) - вектор параметрів, що характеризує конфігурацію (структуру) системи; C = (c_1, ..., c_r) вектор параметрів режиму обробки; Y=(y_1 ,..., y_M) — вектор характеристик системи, зв'язаний з параметрами задач Θ, конфігурації S і режиму обробки З залежністю Y=$F(\Theta, S, 3)$; S=$\{S_i\}$ - множина можливих конфігурацій обчислювальних систем; C = $\{C_J\}$ — множина можливих режимів обробки.

На відміну від задачі аналізу, спрямованої на визначення характеристик системи Y по заданих параметрах X, завдання синтезу полягає у визначенні параметрів конфігурації S і режиму обробки C, що відповідають параметрам робочого навантаження Θ і характеристикам системи Y, заданим у виді (7.1), (7.2). Для задач синтезу характерні два наступних моменти. По-перше, передбачається наявність моделі Y=F(Θ, S, C), що встановлює залежність характеристик системи від її параметрів. По-друге, задача синтезу являє собою оптимізаційну задачу і припускає використання методу оптимізації, що відповідає виду цільової функції (7.1) і обмеженням (7.2). Метод оптимізації повинний гарантувати визначення глобального оптимуму цільової функції E = Φ(Y), визначеної на множині конфігурацій S і режимів обробки C.

Складність задачі синтезу обчислювальної системи обумовлена числом варійованих параметрів, що описують конфігурацію і режим функціонування системи, і областю варіювання параметрів. При загальній постановці задачі синтезу, коли множині конфігурацій S і режимів обробки C містять у собі всі мислимі варіанти побудови систем (одномашинні і багатомашинні, мультіпроцесорні і мережні) і різні способи керування задачами, даними і завданнями, складність задачі синтезу перевершує можливості методів моделювання й оптимізації. Тому в загальній постановці задача синтезу обчислювальних систем виявляється нерозв'язною. Для рішення задачі синтезу її спрощують поділом на послідовність етапів, на кожнім з який виявляються окремі аспекти організації системи.

1. Виходячи з призначення системи (клас розв'язуваних задач, технологія обробки даних, вимоги до продуктивності й умови роботи) і стану елементної бази визначається клас обчислювальної системи: одномашинна система, мультипроцессорний комплекс, локальна мережа й ін. При цьому аналізується ефективність систем різних класів і вибирається клас, що щонайкраще задовольняє призначенню системи.
2. В обраному класі систем синтезується архітектура системи: визначається склад пристроїв, їхні функціональні можливості і технічні характеристики, типи інтерфейсів і структура зв'язків між пристроями, щонайкраще відповідають призначенню системи.
3. Визначається режим обробки даних і його параметри (склад і функції системних процесів, алгоритми розподілу ресурсів між завданнями і задачами, типи і діапазони пріоритетів і ін.). Цим установлюються функції керуючих програм операційної системи і склад системного програмного забезпечення.

Навіть при поділі задачі синтезу на три чи більші числа етапів синтез, зв'язаний з кожним етапом, не вдається звести до єдиної математичної процедури — задачі математичного програмування. Це обумовлено двома основними причинами. По-перше, синтез зв'язаний з різнотипними параметрами: одні є кількісними, а інші — якісними, тобто ознаками типу структури, пристроїв, пам'яті, інтерфейсів, способів керування процесами й ін. Тому синтез зводиться до задач чисельного математичного програмування, а також до комбінаторних задач на сполученнях, відносинах і т.д. Об'єднання різнотипних задач в одну задачу оптимізації виявляється не результативним через різнотипність обчислювальних процедур, що повинні використовуватися в процесі оптимізації. По-друге моделі, що маються в розпорядженні дослідників носять, як правило, локальний характер, відтворюючи властивості досить вузького класу структурних рішень і режимів функціонування. Об'єднання

простих моделей у складну приводить до розривів функцій, не опуклим залежностям, що не тільки утрудняє, але практично виключає можливість застосування методів оптимізації.

З цих причин при синтезі систем прагнуть по можливості зменшувати розмірність задач шляхом поділу задачі синтезу на послідовність етапів, що зводяться до чисто комбінаторних, чи задачам чисельної оптимізації. При цьому проектування системи ведеться зверху вниз — від найбільш загальних рішень, зв'язаних із системою в цілому, до часткових рішень, що відноситься до окремих підсистем і їхніх частин. При розвязку багатьох задач синтезу приходиться використовувати дуже прості моделі функціонування системи і її складових, що приблизно представляють залежності між характеристиками і параметрами. У цих умовах вибір проектних рішень здійснюється на основі досвіду й інтуїції розроблювачів. Таким чином, синтез обчислювальних систем зводиться до розв’язку значного числа взаємозалежних задач вибору способів організації і визначення параметрів проектованої системи в різних аспектах її організації й у відношенні до різних підсистем і елементів. При цьому використовуються як формальні, так і евристичні методи, причому на останні приходиться значне число проектних задач, що виходять за рамки можливостей відомих методів теорії обчислювальних систем

4. СИСТЕМИ ОБРОБКИ ДАНИХ

4.1. Способи побудови і класифікація

Система обробки даних (СОД) — сукупність технічних засобів і програмного забезпечення, призначена для інформаційного обслуговування користувачів і технічних об'єктів. До складу ***технічних засобів*** входить устаткування для введення, зберігання, перетворення і виведення даних, зокрема ЕОМ, пристрою сполучення ЕОМ з об'єктами, апаратура передачі даних і лінії зв'язку. ***Програмне забезпечення*** (програмні засоби) – сукупність програм, що реалізовують покладені на систему функції. Функції СОД полягають у виконанні необхідних актів обробки даних: введення, зберігання, перетворення і виведення. Прикладами СОД є обчислювальні системи для вирішення наукових, інженерно-технічних, планово-економічних і обліково-статистичних задач, автоматизовані системи управління підприємствами і галузями народного господарства, системи автоматизованого і автоматичного управління технологічним устаткуванням і технічними об'єктами, інформаційно-вимірювальні системи і ін.

Основа СОД — це технічні засоби, оскільки їх продуктивністю і надійністю найбільшою мірою визначається ефективність СОД.

Одномашинні СОД. Історично першими і дотепер широко поширеними є одномашинні СОД, побудовані на базі єдиної ЕОМ з традиційною однопроцесорною структурою. На теперішній час накопичений значний досвід проектування і експлуатації такої СОД, і тому створення їх, включаючи розробку програмного забезпечення, не викликає принципових труднощів. Проте продуктивність і надійність існуючого парку ЕОМ виявляється задовільною лише для обмеженого застосування, коли потрібна відносно невисока продуктивність і допускається простої. Підвищення продуктивності і надійності ЕОМ забезпечується в основному за рахунок вдосконалення елементно-технологічної бази. При будь-якому рівні технології не можна забезпечити абсолютну надійність елементної бази, і тому для одномашинної СОД не можна виключити можливість втрати працездатності. Таким чином, одномашинні СОД лише частково задовольняють потребу в автоматизації обробки даних.

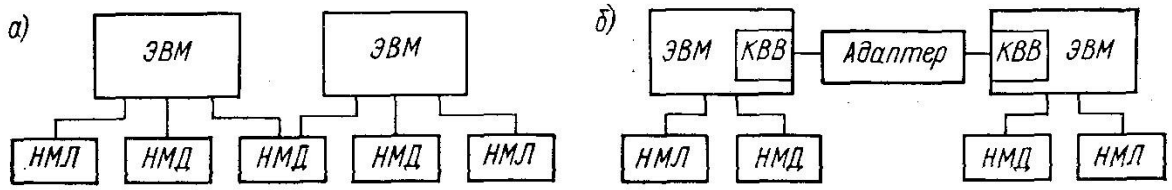


Рисунок 4.1 – Багатомашинний обчислювальний комплекс з непрямим (а) і прямим (б) зв'язком між ЕОМ

Обчислювальні комплекси. Починаючи з 60-х років для підвищення надійності і продуктивності СОД декілька ЕОМ зв'язувалися між собою, утворюючи ***багатомашинний обчислювальний комплекс.***

У ранніх багатомашинних комплексах зв'язок між ЕОМ забезпечувався через загальні зовнішні пристрої, що запам'ятовують, — накопичувачі на магнітних дисках (НМД) або магнітних стрічках (НМС) (рис. 4.1,а), тобто за рахунок доступу до загальних наборів даних. Такий зв'язок називається непрямим і виявляється ефективною тільки у тому випадку, коли ЕОМ взаємодіють достатньо рідко, наприклад при відмові однієї з ЕОМ, або в моменти початку і закінчення обробки даних. Більш оперативна взаємодія ЕОМ досягається за рахунок ***прямого зв'язку*** через адаптер, що забезпечує обмін даними між каналами введення — виведення (КВВ) двох ЕОМ (рис. 4.1, б) і передачу сигналів переривання. За рахунок цього

створюються добрі умови для координації процесів обробки даних і підвищується оперативність обміну даними, що дозволяє вести паралельно процеси обробки і істотно збільшувати продуктивність СОД. В даний час багатомашинні обчислювальні комплекси широко використовуються для підвищення надійності і продуктивності СОД.

У багатомашинних обчислювальних комплексах взаємодія процесів обробки даних забезпечується тільки за рахунок обміну сигналами переривання і передачі даних через адаптери канал — канал, або загальні зовнішні запам'ятовуючі пристрої. Кращі умови для взаємодії процесів — коли всі процесори мають доступ до всього об'єму даних, що зберігаються в оперативних запам’ятовуючих пристроях (ОЗУ), і можуть взаємодіяти зі всіма периферійними пристроями комплексу. Обчислювальний комплекс, що містить декілька процесорів із загальною оперативною пам'яттю і периферійними пристроями, називається ***багатопроцесорним.*** Принцип побудови таких комплексів ілюструється (рис.4.2). Процесори, модулі оперативної пам'яті (МП) і канали введення—виведення, до яких підключені периферійні пристрої (ПП), об'єднуються в єдиний комплекс за допомогою засобів комутації, що забезпечують доступ кожного процесора до будь-якого модуля оперативної пам'яті і каналу введення—виведення, а також можливість передачі даних між останніми. У багатопроцесорному комплексі відмови окремих пристроїв впливають на працездатність СОД у меншій мірі, ніж в багатомашинному, тобто багатопроцесорні комплекси володіють більшою стійкістю до відмов. Кожен процесор має безпосередній доступ до всіх даних, що зберігаються в загальній оперативній пам'яті, і до периферійних пристроїв, що дозволяє паралельно обробляти не тільки незалежні задачі, але і блоки однієї задачі.

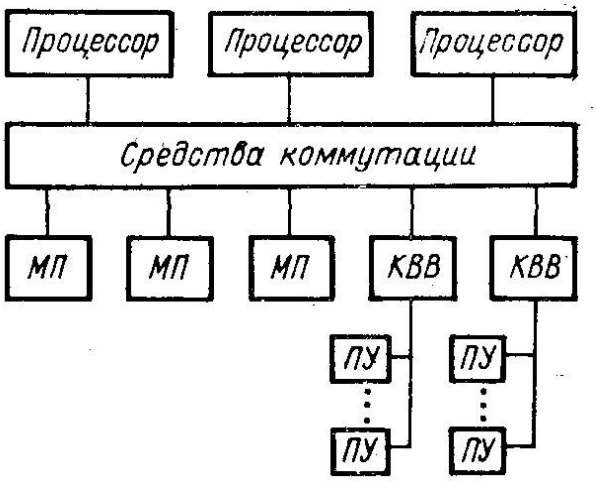


Рисунок 4.2 – Багатопроцесорний обчислювальний комплекс

Багатомашинні і багатопроцесорні обчислювальні комплекси розглядаються як базові засоби для створення СОД різного призначення. Тому до складу обчислювального комплексу прийнято включати тільки технічні засоби і загальносистемне (базове), але не прикладне програмне, забезпечення, пов'язане з конкретною областю застосування комплексу. Таким чином, ***обчислювальний комплекс*** — сукупність технічних засобів, що включають декілька ЕОМ, або процесорів, і загальносистемного (базового) програмного забезпечення.

Обчислювальні системи. СОД, налаштована на розв’язок задач конкретної області застосування; називається ***обчислювальною системою***. Обчислювальна система включає технічні засоби і програмне забезпечення, орієнтовані на розв’язок певної сукупності задач. Існує два способи орієнтації:

По-перше, обчислювальна система може будуватися на основі ЕОМ, або обчислювального комплексу загального застосування і орієнтація системи забезпечується за рахунок програмних засобів — прикладних програм і, можливо, операційної системи.

По-друге, орієнтація на заданий клас задач може досягатися за рахунок використовування спеціалізованих ЕОМ і обчислювальних комплексів. В цьому випадку вдається при помірних витратах устаткування досягти високої продуктивності. Спеціалізовані обчислювальні системи найбільш ширше використовуються при розв'язку задач векторної і матричної алгебри, а також пов'язаних з інтегруванням диференціальних рівнянь, обробкою зображень, розпізнаванням образів і т.д.

Обчислювальні системи, побудовані на основі спеціалізованих комплексів, почали інтенсивно розроблятися з кінця 60-х років. У таких системах використовувалися процесори із спеціалізованими системами команд і конфігурація комплексів жорстко орієнтувалася на конкретний клас задач. Потім почалися дослідження і розробки адаптивних обчислювальних систем, що гнучко пристосовуються до вирішуваних задач. Адаптація обчислювальної системи з метою пристосування її до структури алгоритму, що реалізується досягається за рахунок зміни конфігурації системи. При цьому з'єднання між процесорами, а також модулями пам'яті і периферійними пристроями встановлюються динамічно відповідно до потреб задач, шо обробляються системою у даний момент часу. У зв'язку з цим ***адаптивні обчислювальні системи*** інакше називаються ***системами з динамічною структурою***. За рахунок адаптації досягається висока продуктивність в широкому класі задач і забезпечується стійкість системи до відмов. Тому адаптивні системи розглядаються як один з перспективних напрямів розвитку систем обробки даних.

Системи телеобробки. Вже первинне застосування СОД для управління виробництвом, транспортом і матеріально-технічним постачанням показало, що ефективність систем можна значно підвищити, якщо забезпечити введення даних в систему безпосередньо з місць їх появи і видачу результатів обробки до місць їх використовування. Для цього необхідно зв'язати СОД і робочі місця користувачів за допомогою каналів зв'язку. Системи, призначені для обробки даних, що передаються по каналах зв'язку, називаються ***системами телеобробки даних***.

Склад технічних засобів системи телеобробки даних представлений на рис. 4.3. Користувачі (абоненти) взаємодіють з системою за допомогою терміналів (абонентних пунктів), що підключаються через канали зв'язку до засобів обробки даних — ЕОМ, або обчислювальному комплексу. Дані передаються по каналах зв'язку у формі повідомлень — блоків даних, несучих у собі окрім власне даних службову інформацію, необхідну для управління процесами передачі і захисту даних від спотворень. Програмне забезпечення систем телеобробки містить спеціальні засоби, необхідні для управління технічними засобами, встановлення зв'язку між ЕОМ і абонентами, передачі даних між ними і організації взаємодії користувачів з програмами обробки даних.

Телеобробка даних значно підвищує оперативність інформаційного обслуговування користувачів і разом з цим дозволяє створювати великомасштабні системи, що забезпечують доступ широкого кола користувачів до даних і процедур їх обробки.

Обчислювальні мережі. Із зростанням масштабів застосування електронної обчислювальної техніки в наукових дослідженнях, проектно-конструкторських роботах, управлінні виробництвом і транспортом і інших областях стала очевидна необхідність об'єднання СОД, що обслуговують окремі підприємства і колективи. Об'єднання розрізненої СОД забезпечує доступ до даних і процедур їх обробки для всіх користувачів, зв'язаних загальною сферою діяльності. Так, експериментальні дані, одержані групою дослідників, можуть використовуватися при проектно-конструкторських роботах, результати проектування – при технологічній підготовці виробництва, результати випробувань і експлуатації виробів – для вдосконалення конструкцій і технології виробництва і т.д. Щоб об'єднати територіально розрізнені СОД в єдиний комплекс, необхідно, *по-перше*, забезпечити можливість обміну даними між СОД, зв'язавши відповідні ЕОМ і обчислювальні комплекси каналами передачі даних, і, *по-друге*, забезпечити системи програмними засобами, які

дозволяють користувачам однієї системи звертатися до інформаційних, програмних і технічних ресурсів інших систем.

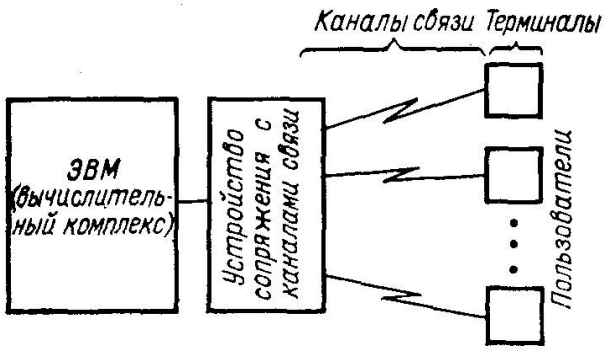


Рисунок 4.3 – Система телеобробки даних

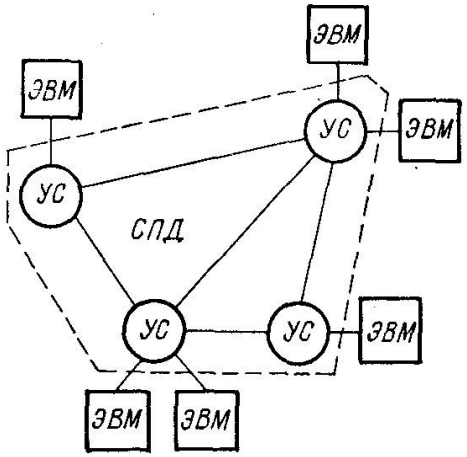


Рисунок 4.4 – Обчислювальна мережа

В кінці 60-х років був запропонований спосіб побудови *обчислювальних мереж,* які об'єднують ЕОМ (обчислювальні комплекси) за допомогою базової мережі передачі даних. Структура обчислювальної мережі у загальних рисах представлена на рис.4.4. Ядром є ***базова мережа передачі даних*** (СПД), яка складається з каналів і вузлів зв'язку (ВЗ). Вузли зв'язку приймають дані і передають їх у напрямі, що забезпечує доставку даних абоненту. ЕОМ підключаються до вузлів базової мережі передачі даних, чим забезпечується можливість обміну даними між будь-якими парами ЕОМ. Сукупність ЕОМ, об'єднаних мережею передачі даних, утворює ***мережу ЕОМ***. До ЕОМ безпосередньо, або за допомогою каналів зв'язку підключаються термінали, через які користувачі взаємодіють з мережею. Сукупність терміналів і засобів зв'язку, що використовуються для підключення терміналів до ЕОМ, утворює ***термінальну мережу***. Таким чином, *обчислювальна мережа є композицією базової мережі передачі даних, мережі ЕОМ і термінальної мережі.* Така обчислювальна мережа називається ***глобальною або розподіленою*** (надалі — «обчислювальна мережа», на відміну від локальної). Обчислювальні мережі використовуються для об'єднання ЕОМ, що знаходяться на значній відстані одна від одної в межах регіону, країни або континенту.

У обчислювальній мережі всі ЕОМ оснащуються спеціальними програмними засобами для мережевої обробки даних. На мережеве програмне забезпечення покладається широкий комплекс функцій: *управління апаратурою сполучення і каналами зв'язку; встановлення з'єднань між взаємодіючими процесами і ЕОМ; управління процесами передачі даних; введення і виконання завдань від видалених терміналів; доступ програм до наборів даних, розміщених у видалених ЕОМ, і ін.* До мережевого програмного забезпечення ставляться наступні вимоги:

- збереження працездатності мережі при зміні її структури унаслідок виходу з ладу окремих ЕОМ, каналів і вузлів зв'язку;
- можливість роботи ЕОМ з терміналами різних типів і взаємодії різнотипних ЕОМ.

Функції, що покладаються на мережеве програмне забезпечення, відрізняються високим рівнем складності і реалізуються з використанням спеціально розроблених методів управління процесами передачі і обробки даних.

Обчислювальні мережі – найефективніший спосіб побудови великомасштабної СОД. Використовування обчислювальних мереж дозволяє автоматизувати управління галузями виробництва, транспортом і матеріально-технічним постачанням в масштабі регіонів і країни в цілому. За рахунок концентрації в мережі великих об'ємів даних і загальнодоступності засобів обробки значно поліпшується інформаційне обслуговування наукових досліджень, підвищується продуктивність праці інженерно-технічних працівників і якість

адміністративно-управлінської діяльності. Крім того, об'єднання ЕОМ в обчислювальні мережі дозволяє істотно підвищити ефективність їх використовування. Як показує практика, вартість обробки даних в обчислювальних мережах, принаймні в півтора рази менше, ніж при використовуванні автономних ЕОМ.

Локальні обчислювальні мережі. З кінця 70-х років у сфері обробки даних широке поширення разом з ЕОМ загального призначення набули міні- і мікро-ЕВМ і почали застосовуватися персональні ЕОМ. При цьому для обробки даних в рамках одного підприємства, або його підрозділу використовувалося велика кількість ЕОМ, кожна з яких обслуговувала невелику групу користувачів, а мікро-ЕВМ і персональні ЕОМ — окремих користувачів. В той же час колективний характер праці вимагав оперативного обміну даними між користувачами, тобто об'єднання ЕОМ в єдиний комплекс. В кінці 70-х років був розроблений ефективний спосіб об'єднання ЕОМ, розташованих на незначній відстані один від одного — в межах однієї будівлі, або групи сусідніх будівель, за допомогою моноканалу (послідовного інтерфейсу) — Лол бальні обчислювальні мережі.

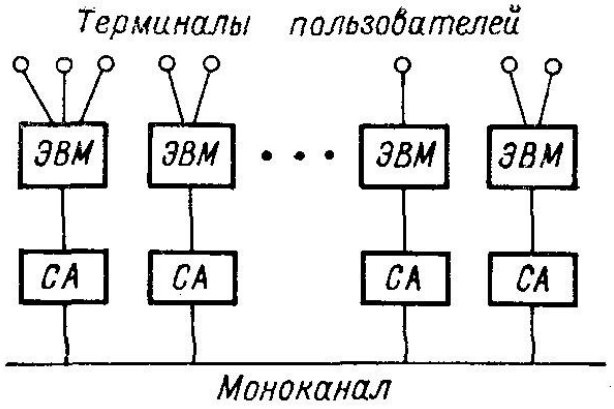


Рисунок 4.5 – Локальна обчислювальна мережа

Локальна обчислювальна мережа (ЛОМ) — сукупність блізькорозташованих ЕОМ, які зв'язані послідовними інтерфейсами і оснащені програмними засобами, що забезпечують інформаційну взаємодію між процесами в різних ЕОМ. Типова структура ЛОМ зображена на рис. 4.5. Сполучаються ЕОМ за допомогою моноканалу – єдиного для всіх ЕОМ мережі каналу передачі даних. У моноканалі найбільш широко використовуються скручена пара дротів (вита пара), коаксіальний кабель, або волоконно-оптична лінія. Довжина моноканалу не перевищує звичайно декількох сотень метрів. При цьому пропускна спроможність моноканалу достатня для забезпечення інформаційного зв'язку між десятками ЕОМ. ЕОМ сполучаються з моноканалом за допомогою мережевих адаптерів (МА), інакше контроллерів, що реалізовують операції введення — виведення даних через моноканал. Наявність в мережі єдиного каналу для обміну даними між ЕОМ істотно спрощує процедури встановлення з'єднань і обміну даними між ЕОМ. Тому мережеве програмне забезпечення ЕОМ виявляється простішим, ніж в обчислювальних мережах, що містять мережу передачі даних, і легко вбудовується в ЕОМ. Внаслідок цього локальні обчислювальні мережі є ефективним засобом побудови складної СОД на основі мікро- і міні-ЕОМ.

Локальні обчислювальні мережі отримали широке застосування в системах автоматизації проектування і технологічної підготовки виробництва, системах управління виробництвом, транспортом, постачанням і збутом (установчих системах), а також в системах автоматичного управління технологічним устаткуванням, створених на основі мікро- і міні-ЕОМ, зокрема в гнучких виробничих системах.

Класифікація СОД. Класифікуються СОД залежно від способу побудови (Рис. 4.6). СОД, побудовані на основі окремих ЕОМ, обчислювальних комплексів і систем, утворюють клас ***зосереджених (централізованих) систем***, в яких вся обробка реалізується ЕОМ,

обчислювальним комплексом, або спеціалізованою системою. Системи телеобробки і обчислювальні мережі відносяться до класу ***розподілених систем***, в яких процеси обробки даних розосереджені по багатьох компонентах. При цьому системи телеобробки вважаються розподіленими в деякій мірі умовно, оскільки основні функції обробки даних тут реалізуються централізовано — в одній ЕОМ, або обчислювальному комплексі.

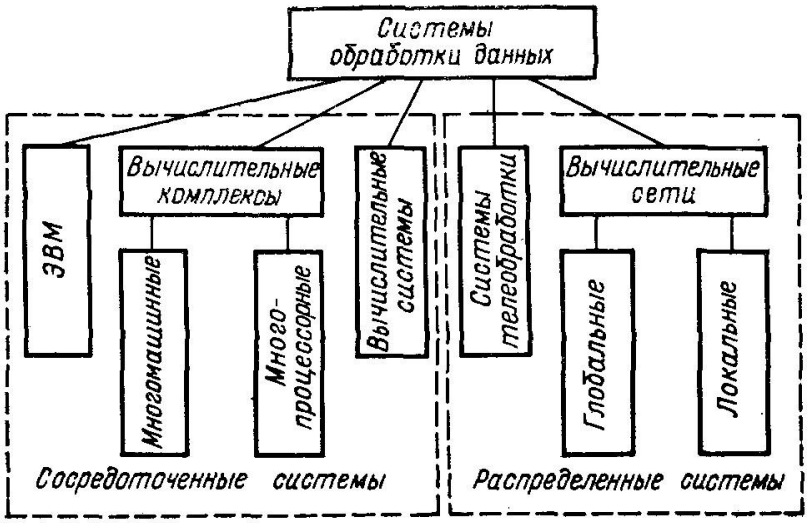


Рисунок 4.6 – Класифікація СОД

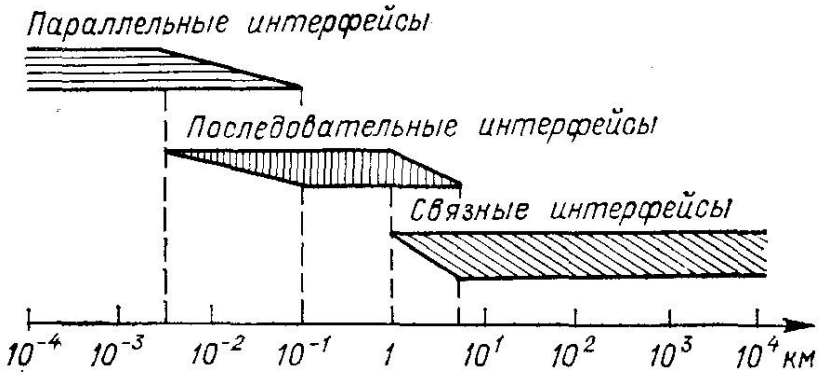


Рисунок 4.7 – Характеристики інтерфейсів

Істотний вплив на організацію СОД роблять технічні можливості засобів, що використовуються для сполучення (комплексування) ЕОМ. Основним елементом сполучення є *інтерфейс*, що визначає число ліній для передачі сигналів і даних і спосіб (алгоритм) передачі інформації по лініях. Всі інтерфейси, що використовуються в обчислювальній техніці і зв'язку, поділяються на три класи: ***паралельні, послідовні і зв'язні*** (рис. 4.7). ***Паралельний інтерфейс*** складається з великої кількості ліній, дані по яких передаються в паралельному коді — звично у вигляді 8—128-розрядних слів. Паралельні інтерфейси мають велику пропускну спроможність, як правило, 10^6-10^8 біт/с. Такі великі швидкості передачі даних забезпечуються за рахунок обмеженої довжини інтерфейсу, яка звичайно складає від декількох метрів до десятків метрів і в окремих випадках досягає сотні. ***Послідовний інтерфейс*** складається, як правило, з однієї лінії, дані по якій передаються в послідовному коді. Пропускна спроможність послідовних інтерфейсів звичайно складає 10^5-10^7 біт/с при довжині лінії від десятків метрів до кілометра. ***Зв'язні інтерфейси*** містять канали зв'язку, робота яких забезпечується апаратурою передачі даних, що підвищує (в основному за допомогою фізичних методів) достовірність передачі. Зв'язні інтерфейси забезпечують передачу даних на будь-які відстані, проте з невеликою швидкістю — в межах від 10^3 до 10^5

біт/с. Застосування зв'язних інтерфейсів економічно виправдовується на відстані, не менших кілометра.

У зосереджених системах застосовуються в основному паралельні інтерфейси, що використовуються для сполучення пристроїв і побудови багатомашинних і багатопроцесорних комплексів, і лише в окремих випадках, частіше для підключення периферійних пристроїв, застосовуються послідовні інтерфейси. Паралельні інтерфейси забезпечують передачу сигналів переривання, окремих слів і блоків даних між ЕОМ, що сполучаються, і пристроями. У розподілених системах через значних відстаней між компонентами застосовуються послідовні і зв'язні інтерфейси, які виключають можливість передачі сигналів переривання між пристроями, що сполучаються, і вимагають представлення даних у вигляді повідомлень, що передаються за допомогою операцій введення — виведення. Відмінність способів представлення даних в паралельних, послідовних і зв'язних інтерфейсах і в пропускній спроможності інтерфейсів істотно впливає на організацію обробки даних і, отже, програмного забезпечення СОД.

4.2. Склад і функціонування

Системи обробки даних будуються з технічних і програмних засобів, що істотно розрізняються за своєю природою. Тому СОД прийнято розглядати як сукупність двох складових: технічних засобів і програмного забезпечення. Функціонування СОД визначається взаємодією програмних і технічних засобів, внаслідок чого властивості системи проявляться як сукупні властивості технічних і програмних засобів.

Технічні засоби. Основу СОД складають технічні засоби — устаткування, призначене для введення, зберігання, перетворення і виведення даних. Склад технічних засобів визначається ***структурою (конфігурацією) СОД***, тобто тим, з яких частин (елементів) складається система і яким чином ці частини зв'язані між собою. Математична форма представлення структури — ***граф***, вершини якого відповідають елементам системи, а ребра (дуги) — зв’язкам між ними. Інженерна форма представлення структури — ***схема***. Таким чином, схема і граф тотожні за змістом і різні формою, В схемі для зображення елементів використовуються різні геометричні фігури, а для зображення зв'язків — лінії багатьох типів. За рахунок цього схема має велику в порівнянні з графом наочність. Основні елементи структури СОД — пристрої: процесори, пристрої запам'ятовують, введення — виведення, сполучення з об'єктами і ін. Пристрої зв'язуються за допомогою інтерфейсів, що включають сукупність ліній або каналів передачі даних (ліній зв'язку).

Приклад структури, представленої на рівні пристроїв, приведений на рис. 4.8. До складу даного комплексу входять дві ЕОМ, кожна з яких забезпечена трьома каналами введення — виведення *МК.0, СК1* і *СК2,* двома накопичувачами на магнітних дисках *НМД1* і *НМД2* і дисплеями *Д1* і *Д2,* підключеними через контролер *КД* до мультиплексного каналу *МК0.* Машини пов'язані із загальним для них набором зовнішніх пристроїв, що запам'ятовують, — накопичувачами на магнітних дисках *НМД3* і *НМД4* і магнітних стрічках *НМЛ1—НМЛ4,* які підключені до селекторних каналів *СК2* через відповідні контролери *КНМД* і *КНМЛ.* До ЕОМ підключені мультиплексори передачі даних *МПД1* і *МПД2,* кожний з яких обслуговує чотири канали зв'язку *КС1—КС4* і *КС5—КС8.* На рисунку лініями представлені наступні інтерфейси: інтерфейс прямого управління, що сполучає процесори *ЭВМ1* і *ЭВМ2;* інтерфейси оперативної пам'яті, що зв’язують оперативну пам'ять з процесором і каналами введення — виведення *МК0, СК1* і *СК2;* інтерфейси введення — виведення, що зв’язують канали введення — виведення з контролерами пристроїв, що запам'ятовують, і пристроїв введення — виведення; малі інтерфейси, за допомогою яких накопичувачі і пристрої введення — виведення підключаються до відповідних контролерів.

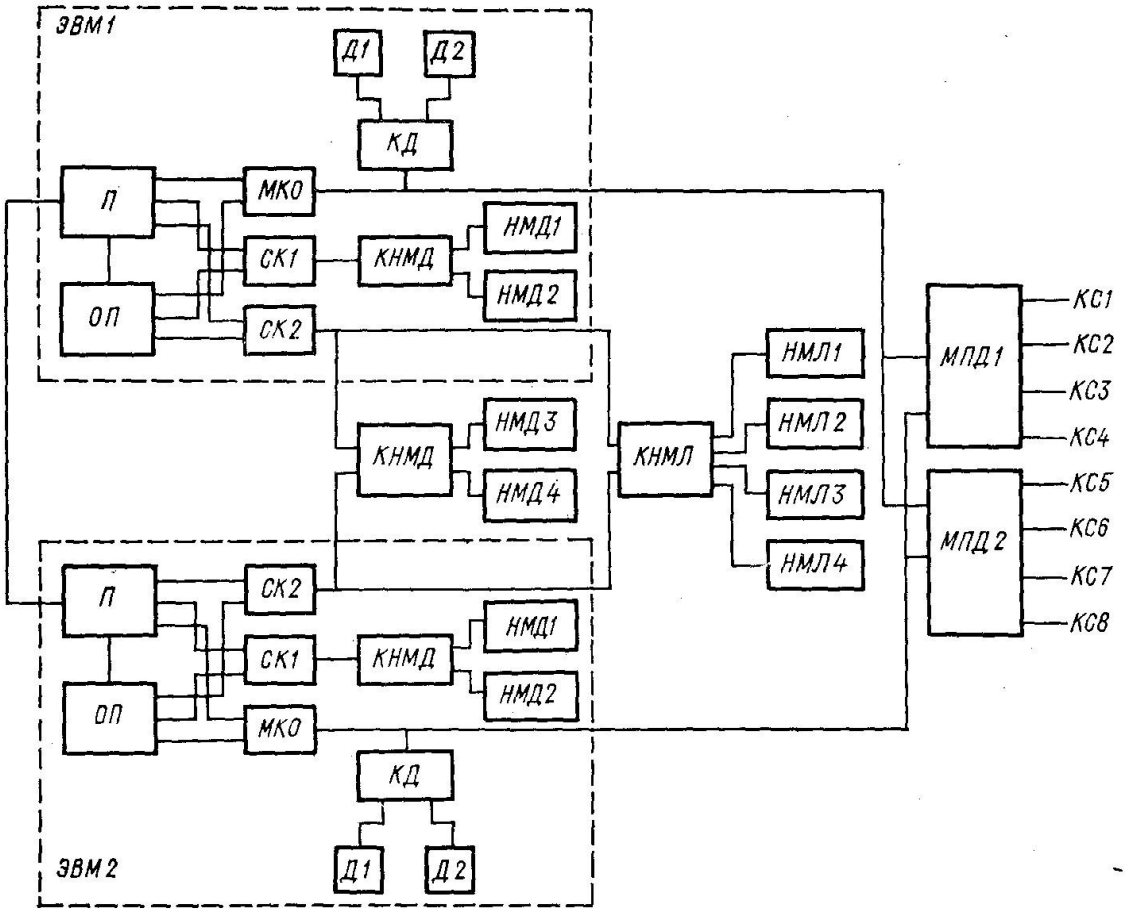


Рисунок 4.8 – Двомашинний обчислювальний комплекс. П – процесор; *ОП* – оперативна пам'ять; МК – мультиплексний канал; КНМД — контролер НМД; КНМЛ — контролер НМЛ; *КД* — контролер дисплеїв; МПД — мультиплексор передачі даних; КС — канал зв'язку; Д — дисплей

Структура складних систем при представленні її на рівні пристроїв може виявитися настільки складною, що втрачає осяйність і виходить за рамки можливостей методів дослідження, що використовуються при аналізі і синтезі систем. У таких випадках структура описується на більш високому рівні, коли у якості елементів виступають ЕОМ, багатопроцесорні комплекси і складні підсистеми, які представляються однією вершиною графа. Таким чином, елемент структури СОД — це перш за все зручне поняття, але не фізична властивість об'єкту. Головна вимога до зображення структури — інформативність.

Структура СОД дає загальне уявлення про склад технічних засобів і зв'язків між ними. Додаткові відомості про технічні засоби даються у формі специфікації, де для кожного елементу структури і кожного типу зв'язків між елементами вказується: найменування елементу, приведене на структурній схемі; тип пристрою, відповідного елементу структурної схеми; технічні характеристики пристрою або засобу зв'язку (продуктивність, місткість пам'яті, пропускна спроможність).

У плані зв'язку з процесами обробки даних технічні засоби розглядаються як сукупність ресурсів двох типів: пристроїв і пам'яті. ***Пристрій*** — ресурс, що використовуваний для перетворення і введення — виведення даних, що поділяється між процесами (задачами) в часі. Приклади пристроїв — процесори, канали введення — виведення, периферійні пристрої (введення — виведення і що зовнішні запам'ятовуючі), канали передачі даних. У кожен момент часу пристрій використовується одним процесом, реалізовуючи відповідні операції: перетворення. або введення — виведення даних. *Основна характеристика пристрою* —

продуктивність, що визначається числом операцій, виконуваних в секунду, або пропускна спроможність, що визначається кількістю одиниць інформації (байтів), переданих в секунду.
Пам'ять — ресурс, що використовується для зберігання даних і, що поділяється між процесами за об'ємом і часом. Приклади — оперативна пам'ять і накопичувачі на магнітних дисках. Основна характеристика пам'яті — *місткість*, що визначається граничною кількістю інформації в пам'яті. У одній пам'яті одночасно можуть розміщуватися дані, що відносяться до декількох процесів. Накопичувач на магнітних дисках містить два ресурси, будучи одночасно пам'яттю певної місткості і пристроєм, що обслуговує операції введення — виведення даних.

Таким чином, склад технічних засобів визначає номенклатуру ресурсів, що використовуються для зберігання, введення — виведення і перетворення даних. Конфігурація зв'язків між пристроями визначає шляхи передачі даних в системі і порядок доступу процесів до пристроїв і даних, що зберігаються в пам'яті. Технічні засоби СОД реалізують елементарні операції введення — виведення і обробки даних. Необхідний набір функцій, визначуваних призначенням СОД, забезпечується сукупністю програм — програмним забезпеченням СОД.

Програмне забезпечення СОД будується за багаторівневим, ієрархічним принципом. Основні процеси обробки даних описуються в термінах операцій над математичними і логічними елементами даних, що вводяться проблемно- і процедурноорієнтованими мовами програмування. Ці операції за допомогою програмних засобів нижчих рівнів інтерпретуються як простіші операції і врешті-решт зводяться до операцій, що реалізуються технічними засобами СОД.

Приклад багаторівневої реалізації функцій в СОД приведений на рис. 4.9. Технічні засоби СОД забезпечують реалізацію елементарних функцій — операцій введення, зберігання, перетворення і виведення даних, які виконуються за допомогою схем і засобів мікропрограмного управління. Функції, що реалізуються технічними засобами, відносяться до першого, нижчого, рівня ієрархії. Функції вищих рівнів складності забезпечуються програмним забезпеченням СОД, що включає операційну систему і прикладне програмне забезпечення.

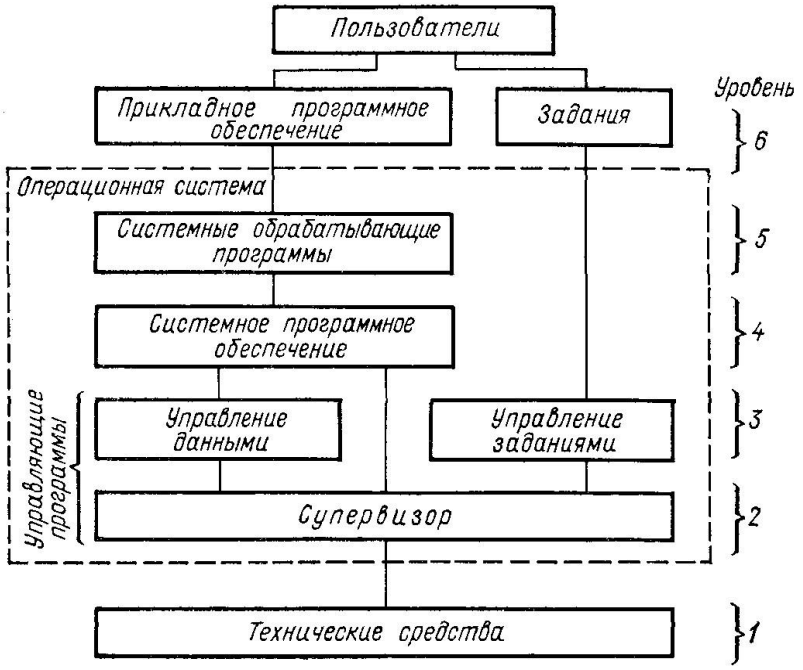


Рисунок 4.9 – Багаторівнева організація СОД

Операційна система (ОС) — сукупність програм, призначених для управління роботою СОД і реалізації наймасовіших процедур взаємодії з користувачами, введення —

виведення, зберігання і перетворення даних. Управління роботою СОД зводиться до управління процесами і ресурсами, що забезпечують ефективне використання устаткування СОД і необхідну якість обслуговування користувачів. Функції управління роботою СОД реалізуються ***управляючими програмами*** **ОС**, що включають в свій склад супервізор, програми управління завданнями і даними. ***Супервізор*** контролює стан всіх технічних засобів і процесів (задач) і керує ними, забезпечуючи необхідний режим обробки даних шляхом розподілу процесів у просторі та часі. Супервізор виділяє задачам області (розділи) пам'яті і пристрої введення — виведення, ініціює виконання процесором програм, починає операції введення – виведення і обробляє сигнали переривання, що відзначають закінчення операцій введення – виведення і особливі ситуації, що виникають при виконанні програм і роботі пристроїв.

Програми управління завданнями забезпечують введення і інтерпретацію команд операторів, що управляють роботою СОД, і завдань, що формуються користувачами

СОД. Оператори за допомогою спеціальних команд впливають на порядок функціонування і одержують інформацію про поточний стан СОД. Ці програми інтерпретують завдання у вигляді відповідних дій і забезпечують їх необхідними ресурсами — розділами оперативної і зовнішньої пам'яті, пристроями введення — виведення, наборами даних і ін. Завдання, забезпечені ресурсами, необхідними для їх виконання, утворюють задачі. Управління задачами реалізується супервізором. Для звернення до програм управління завданнями застосовується мова управління завданнями, в термінах якого користувачі і оператори, що управляють роботою системи, записують завдання на виконання робіт в системі.

Програми управління даними забезпечують доступ до наборів даних і

організацію роботи пристроїв введення — виведення. Засоби управління даними настроюють програми на роботу з конкретними наборами даних і пристроями, в яких зберігаються набори, і за рахунок цього створюють можливість при програмуванні задач маніпулювати з даними як з логічними об'єктами, не пов'язаними з конкретними пристроями. Таким чином, управління даними зводиться до сполучення програм з наборами даних і пристроями, а використання цих пристроїв контролюється і координується супервізором.

Функції, що реалізуються управляючими програмами ОС, відносяться до

другого і третього рівня функцій системи (див. рис. 4.9).

Функції ОС розширяються за рахунок засобів ***системного програмного***

забезпечення – програмних засобів телеобробки, управління базами даних, мережевої обробки і ін. Системне програмне забезпечення є основою для побудови прикладного програмного забезпечення і надає користувачу засоби, необхідні для роботи із спеціальними пристроями (наприклад, з апаратурою передачі даних і віддаленими терміналами), або для спеціальної обробки даних. Функції, що реалізуються засобами системного програмного забезпечення, відносяться до четвертого рівня ієрархії.

До п'ятого рівня відносяться функції, що виконуються ***системними***

обробляючими програмами **ОС**. Ці програми включають: транслятори з мов програмування; редактори зв'язків, що забезпечують збирання програмних модулів в програми із заданою структурою; засоби налагоджування програм і переміщення наборів даних з одних носіїв на інші і т.д. Функції, що забезпечуються трансляторами, представляються у вигляді мов програмування: машинно-, процедурно- і проблемно-орієнтованих мов, мов генерації програм введення — виведення і ін.

Прикладне програмне забезпечення — сукупність прикладних програм, що реалізовують функції обробки даних, і які пов'язані з конкретною областю застосування системи. У системах автоматизації проектування радіоелектронної апаратури прикладні програми забезпечують аналіз електронних схем, розміщення електронних елементів по конструктивних одиницях, розводку з'єднань на друкарській платні і т. д.; в автоматизованих

системах управління виробництвом — календарне і оперативне планування виробництва на підприємстві і в низових виробничих підрозділах, облік і аналіз виробничої діяльності і т.д.

Склад прикладних програм визначається призначенням системи.

До програмних засобів СОД примикають ***набори даних***, що розглядаються як особлива складова — інформаційне забезпечення СОД. ***Набори даних*** — сукупність логічно зв'язаних елементів даних, організованих за певними правилами і забезпечених описом, доступним системі програмування (засобам управління даними). Набори даних забезпечуються іменами, за допомогою яких програми звертаються до відповідних наборів і їх елементів. Одні і ті ж набори даних можуть використовуватися багатьма прикладними програмами. Щоб виключити необхідність представлення одних і тих же даних в різній формі, варіантах і поєднаннях, орієнтованих на різні програми, необхідно забезпечити незалежність даних і програм. Це досягається за рахунок організації даних у вигляді спеціальних структур — ***баз*** і ***банків даних***, а також використовуванням сукупності програмних засобів, призначених для вибірки, модифікації і додавання даних, – ***системи управління базами даних***. Організація даних у формі баз забезпечує незалежність прикладних програм від логічної і фізичної організації бази даних, внаслідок чого зміни в програмах не спричиняють за собою змін бази і реорганізація бази даних не вимагає внесення змін в програми, що оперують з даними.

Функціонування СОД. Функціонування СОД представляється у вигляді процесів. ***Процес*** – це динамічний об'єкт, що реалізовує цілеспрямований акт обробки даних. Процеси поділяються на прикладні і системні. ***Прикладні процеси*** реалізують основні функції СОД, задані прикладними програмами, або обробляючими програмами ОС, і ініціюються завданнями користувачів або сигналами, що надходять в СОД із зовнішнього середовища. Приклади прикладних процесів: розв’язок прикладної задачі; редагування, трансляція і збірка програми; сортування набору даних і ін. ***Системні процеси*** реалізують допоміжні функції, що забезпечують роботу СОД. Приклади системних процесів: системне введення; системний висновок; переміщення сторінок у віртуальній пам'яті; робота супервізора і ін. Як правило, системні процеси існують протягом всього періоду роботи СОД — від моменту включення до моменту виключення СОД.

Процес P_i описується трійкою $P_i=<t_i, A_i, T_i>$, де t_i — момент ініціації процесу, A_i — атрибути процесу, що визначають імена джерела, що ініціювало процес, користувача, завдання, режим обробки даних, пріоритет процесу і ін., і T_i — траса процесу. ***Траса процесу*** – послідовність подій, пов'язаних із зміною стану процесу. Траса процесу представляється у вигляді впорядкованої множини подій $T_i = \{s_1, s_2, ..., s_M\}$, що мали місце в моменти часу $t_1, t_2, ..., t_M$, причому $t_1 \le t_2, ..., \le t_M$. До подій відносяться моменти введення завдання, початки і завершення обробки кроків (пунктів) завдання, початки і закінчення виконання процесів в пристроях СОД, початки використовування і звільнення розділів пам'яті, що надаються процесу в запам'ятовуючих пристроях і ін. Кожна подія зв'язується з моментом її виникнення, програмою, що реалізовує процес, і ресурсом, що обслуговує процес. Таким чином, траса характеризує динаміку процесу — розвиток процесу в часі і просторі. Траса може бути представлена у вигляді часової діаграми, на рис. 4.10 представлене виконання програми процесором і зовнішнім пристроєм. Відрізки, виділені на осях жирними лініями, відповідають періодам, коли процесор і зовнішній пристрій зайняті виконанням програми. Дугами позначені інтервали часу, протягом яких процес знаходиться в стані очікування, тобто не обслуговується жодним пристроєм, чекаючи моменту звільнення пристрою.

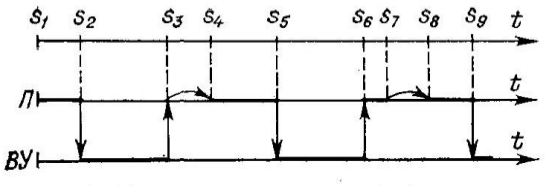


Рисунок 4.10 – Тимчасова діаграма обчислювального процесу

Таким чином, функціонування СОД виражається у формі процесів виконання програм. Процес виконання програми зв'язаний з використанням ресурсів СОД, а також наборів даних і самих програм. Отже, характерною рисою процесу є його одночасний зв'язок і з виконанням програм і з роботою технічних засобів СОД.

Робоче навантаження. Процес функціонування СОД істотно залежить від складу завдань, початкових даних і сигналів, що поступають на вхід СОД. Весь об'єм поступаючої інформації прийнято називати ***робочим навантаженням СОД***. При проектуванні і експлуатації СОД найбільший інтерес представляє потреба завдань в ресурсах: оперативній і зовнішній пам'яті, процесорному часі, пристроях введення — виведення і ін. Тому робоче навантаження, що відноситься до проміжку часу *T,* визначають у вигляді множини характеристик завдань

$$L = \{l_i\} = \{< A_i, \theta_{i1}, \ldots, \theta_{iN} >\} \qquad (4.1)$$

де l_i — опис *i*-го завдання, що встановлює його атрибути A_i і потребу завдання $\theta i1, \ldots, \theta iN$ в ресурсах $1,\ldots, N$. Наприклад, значення $\theta i1$ може визначати місткість області оперативної пам'яті, необхідної завданню, $\theta i2$ – число виконуваних процесором операцій, $\theta i3$ – кількість даних, що вводяться, і т.д.

Кількість завдань, що обробляє СОД за проміжок часу, що дає повне уявлення про робоче навантаження, як правило досить велике. Тому опис робочого навантаження у вигляді (4.1) буде, як правило, громіздким. Для представлення робочого навантаження в компактній формі потреба завдань в ресурсах характеризується середньостатистичними значеннями об'єму ресурсів, що приходиться на одне завдання.

Робоче навантаження залежить від призначення (сфери застосування) СОД і виявляється різним для систем, що оперують з різними класами задач: інженернотехнічними, планово-економічними, обліково-статистичними і ін.

1.3. ХАРАКТЕРИСТИКИ І ПАРАМЕТРИ

При проектуванні СОД необхідно забезпечити якнайповнішу відповідність системи своєму призначенню. Ступінь відповідності системи своєму призначенню називається ***ефективністю (якістю) системи***. Для складних систем, якими є СОД, ефективність неможливо визначити однією величиною, і тому її представляють набором величин, званих ***характеристиками системи.*** Набір характеристик формується так, щоб в своїй сукупності вони давали якнайповніше уявлення про ефективність системи. **Основними характеристиками СОД є продуктивність, час відповіді, надійність і вартість**. На додаток до них використовуються наступні характеристики: габарити, маса, споживана потужність, діапазон робочих температур, ремонтоздатність і ін.

Характеристики залежать від організації системи — структури, складу програмного забезпечення, режиму функціонування системи і ін. Стосовно задач оцінки ефективності організація СОД визначається у вигляді математичних об'єктів, званих ***параметрами системи***. Як параметри використовуються величини, що визначають, наприклад, число і швидкодію пристроїв, місткість пам'яті, робоче навантаження і ін. Поряд з цими величинами як параметри можуть використовуватися такі математичні об'єкти, як множини, графи, алгоритми і ін. У число параметрів включаються всі об'єкти, що характеризують первинні аспекти організації системи і істотно впливають на характеристики.

Таким чином, характеристики визначають властивості системи як цілого, які проявляються в процесі експлуатації системи, і залежні від її організації, що представляється відповідним набором параметрів. У математичному аспекті характеристики можна розглядати як найменування функцій, аргументами яких є параметри.

Можна виділити слідуючи способи оцінки основних характеристик СОД і

набори параметрів, що впливають на характеристики.

Продуктивність. ***Продуктивність***—характеристика обчислювальної потужності системи, визначаюча кількість обчислювальної роботи, що виконує система одиницю часу. В даний час відсутня загальноприйнята методика оцінки продуктивності СОД, що пов'язане в першу чергу з відсутністю одиниць для вимірювання кількості обчислювальної роботи. Тому для оцінки продуктивності використовується широка номенклатура величин – показників продуктивності, які і окремо і в сукупності не задовольняють повною мірою потребам теорії і практики проектування і експлуатації СОД.

Технічні засоби СОД (ЕОМ і периферійне устаткування) володіють продуктивністю поза зв'язком з операційною системою, прикладним програмним забезпеченням і режимом експлуатації системи. Продуктивність технічних засобів оцінюється їх швидкодією — числом операцій, виконуваних ЕОМ і пристроями за секунду. Сукупність значень *V*= (*V1,..., VN*), що визначають швидкодії пристроїв 1,..., *N,* що входять до складу системи, характеризує *номінальну продуктивність* системи. Щоб оцінка номінальної продуктивності була по можливості простій, прагнуть зменшити число складових в наборі (*V1,..., VN*). Це досягається двома способами. По-перше, швидкодія пристроїв, що виконують однакові операції і здатних працювати паралельно, представляють сумарною швидкодією. За рахунок цього номінальна швидкодія може бути представлене як набір значень, а саме сумарна швидкодія процесорів, зовнішньої пам'яті, засобів введення і виведення, наприклад: швидкодія процесорів 1,2 млн. операцій в секунду; швидкодія зовнішньої пам'яті 200 обігу в секунду; швидкість введення б тис. символів в секунду. Подруге, швидкодія зовнішньої пам'яті і підсистем введення — виведення може вважатися неістотним і тоді номінальну продуктивність характеризують одним значенням – сумарною швидкодією процесорів системи.

Номінальна продуктивність характеризує тільки потенційні можливості пристроїв, які не можуть бути використані повністю. Цьому перешкоджає вплив структури зв'язків між пристроями на їх продуктивність, що виявляється в зміні швидкості роботи одних пристроїв при роботі інших. Так, через те що процесор і канали введення — виведення підключені до загальної оперативної пам'яті, збільшення швидкості введення — виведення приводить до зменшення продуктивності процесора; сумарна продуктивність пристроїв введення — виведення, підключених до мультиплексного каналу, обмежена пропускною спроможністю каналу; фактична продуктивність накопичувачів, підключених до блокмультиплексного каналу, менше їх сумарної номінальної швидкодії і т.д. Щоб оцінити вплив першої групи чинників — структури системи на швидкодію пристроїв, використовується спеціальна характеристика — комплексна продуктивність. Комплексна *продуктивність оцінюється* набором швидкодій пристроїв Vi*, .. .,*VN*, *забезпечуваних* при спільній їх роботі, тобто у складі комплексу технічних засобів. З вищеописаних причин комплексна продуктивність нижча номінальною: Vi* ? Vi, …, VN*? VN . *Спосіб* оцінки комплексної продуктивності дотепер не визначений. Один з підходів до її оцінки зводиться до наступного. Деяким чином визначається типова суміш операцій введення, звернення до зовнішньої пам'яті, обробки і виведення даних, на основі якої створюється синтетична штучна програма, що породжує процес із заданою сумішшю операцій. Шляхом прогону синтетичної програми і вимірювання часу її виконання оцінюється комплексна продуктивність системи.

Показником використовування пристрою в процесі роботи системи є *завантаження.* Завантаження *i-го* пристрою визначається відношенням ρi = *Ti* /T, де *Ti* — час, протягом якого пристрій працював, і *T* — тривалість роботи системи. Протягом проміжку часу *T* - *Ti* пристрій простоює. Очевидно, що завантаження ρi ? 1. Якщо завантаження пристроїв l, ..., *N* рівна ρ1, ..., ρN відповідно, та кількість роботи, виконуваної пристроями з швидкодією *V1,..., VN* за диницю часу, рівне c1V1, ..., *cN VN* . *Сукупність значень* c1V1, ..., *cN VN* характеризує продуктивність технічних засобів з урахуванням простоїв, що виникають в процесі

функціонування системи. Таким чином, оцінка фактичної продуктивності системи зводиться до оцінки завантаження пристроїв в конкретних умовах роботи системи.

На завантаження пристроїв істотно впливає режим обробки задач, реалізовуваний управляючими програмами операційної системи. Вплив операційної системи виявляється, наприклад, в наступному. Організація системного введення і виведення зв'язана з використанням процесора і зовнішніх пристроїв, що запам'ятовують, для проміжного зберігання наборів даних, що вводяться і виводяться. В результаті цього частина часу процесора, каналів введення — виведення і зовнішніх пристроїв, що запам'ятовують, витрачається на обслуговування введення—виведення. Така ж ситуація виникає при організації в системі віртуальної пам'яті, режиму розділення часу і забезпечення інших допоміжних функцій.

Щоб оцінити вплив операційної системи на продуктивність технічних засобів СОД, використовується спеціальна характеристика — системна продуктивність. *Системна продуктивність* визначається набором значень ρ1V1, ..., ρN *VN*, в якому завантаження ρ1, ..., ρN визначена при спільній роботі комплексу технічних засобів під управлінням операційної системи. Через вищеописані чинники системна продуктивність нижча комплексною (ρ1V1 ? *V1**, ..., ρN *VN* ? *VN* *) і, отже, нижче номінальною (ρ1V1 ? *V1*, ..., ρN *VN* ? *VN*). В даний час загальноприйнята методика оцінки системної продуктивності відсутня.

Для СОД, що знаходиться в експлуатації або розробляється для конкретного застосування, клас задач повністю визначений принаймні статистично, тобто визначене робоче навантаження СОД. У такому разі продуктивність оцінюється на робочому навантаженні і називається системною *продуктивністю або* коротке —· продуктивністю.

Продуктивність найпростіше оцінюється числом задач, вирішуваних системою за одиницю часу: λ задач/ч. Ця оцінка інформативна тільки для конкретної області застосування СОД і ні про що не свідчить, якщо не визначений клас вирішуваних задач. З цієї причини вона використовується, коли аналізуються варіанти організації однієї СОД, і не може застосовуватися для порівняння СОД, що працює з різними наборами задач.

Розглянемо способи визначення продуктивності на робочому навантаженні для систем, що знаходяться в експлуатації.

Хай за час *T* система завершила обробку $^{3}\eta$ задач (завдань). Тоді

продуктивність системи за час *T* складає

$$\lambda = n/\Phi \qquad (4.2)$$

задач в одиницю часу (наприклад, в годину).

Звичайно задачі поступають на обробку у випадкові моменти часу і час перебування задач в системі залежить від складу суміші (числа і характеристик) задач, одночасно оброблюваних системою. В результаті цього число задач *n,* оброблених системою за час *T,* — випадкова величина і продуктивність λ в інтервалі *T* оцінюється з погрішністю, що має статистичну природу і залежною від випадкової величини *n* і її дисперсії. Із збільшенням тривалості інтервалу значення *n* зростає і погрішність оцінки λ прагне до нуля при *T* > ?.

Інший спосіб визначення продуктивності λ — через середнє значення інтервалу між моментами закінчення обра-ботки задач. У зтом випадку протягом часу *T* реєструються інтервали між моментами завершення обробки задач τ1,...,τ*n* (Рис. 4.11).

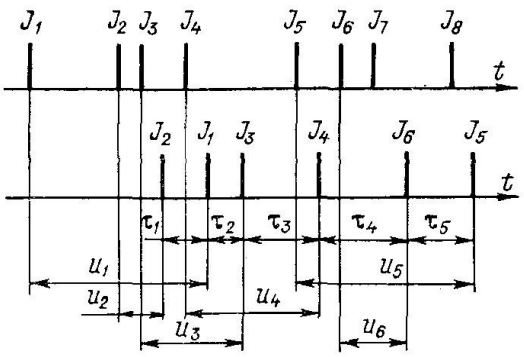


Рисунок 4.11 – Потоки задач на вході і задач на продуктивність і час відповіді

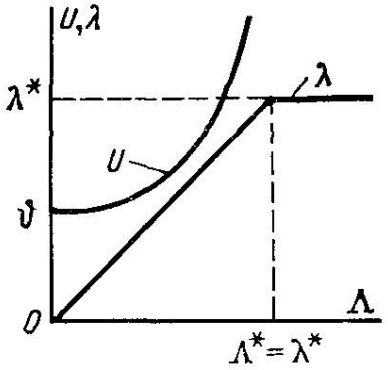


Рисунок 4.12 – Вплив інтенсивності вхідного потоку у виході системи

Розглянемо залежність між двома величинами: середнім числом задач, що поступають в одиницю часу на вхід системи, - інтенсивністю вхідного потоку задач Λ — і середнім числом задач, що покидають систему за одиницю часу, — інтенсивністю вихідного потоку задач λ. Залежність представлена на Рис. 4.12. У області 0?Λ?Λ* інтенсивність вихідного потоку повністю визначається інтенсивністю вхідного потоку: λ=Λ. При Λ?Λ* система через обмеженість ресурсів — числа і швидкодії пристроїв, а також місткості пам'яті — не може протягом одиниці часу обслужити всі завдання, що поступили на обробку, і інтенсивність вихідного потоку λ, досягнувши граничного значення λ*, залишається постійною при будь-яких значеннях Λ?Λ*, причому Λ* = λ*. Значення λ* визначає максимальну продуктивність системи для заданого класу задач і є характеристикою самої системи, не залежної від інтенсивності вхідного потоку задач. Таким чином, продуктивність системи — обмежена зверху величина: 0? λ ? λ*. У області 0?Λ? λ* всі ресурси системи якоюсь мірою недовикористають. У області Λ ? λ* принаймні один ресурс завантажений повністю. Решта ресурсів може бути недовантажені через брак одного або одночасно декількох ресурсів.

На продуктивність найістотніше впливають наступні параметри:

1) число і швидкодія пристроїв, місткість оперативної і зовнішньої пам'яті, із збільшенням яких продуктивність може зростати, а також структура системи і пропускна спроможність зв'язків між елементами системи;

2) режим обробки задач, визначаючий порядок розподілу ресурсів системи між задачами, що поступають на обробку;

3) робоче навантаження, в першу чергу об'їм тих, що вводяться, зберігаються в пам'яті, даних, що виводяться, і число процесорних операцій, необхідних для вирішення задачі.

Оцінка продуктивності у вигляді числа задач, вирішуваних системою за одиницю часу, має сенс тільки для конкретної системи, що працює із заданою безліччю задач. Щоб порівнювати продуктивність різних систем, оброблювальних різні класи задач, продуктивність на робочому навантаженні визначають об'ємом обчислювальної роботи, виконуваної системою за одиницю часу. Таку оцінку представляють набором значень *P1, ..., PN,* складові якого визначають об'єм обробки, введення і виведення даних в одиницю часу. Наприклад, *P1* — число процесорних операцій в секунду, *P2* — число символів, що вводяться за секунду. Як і при оцінці продуктивності λ числом задач в одиницю часу, продуктивність системи в об'ємі робіт *P1, ..., PN* характеризується граничним значенням *P1*, ..., PN*,* що досягається при насиченні системи.

Оцінки продуктивності λ і *P1, ..., PN* зв'язані таким чином. Хай *θ1, ..., θN* — складність обчислень, що характеризується середнім числом операцій типу 1, …, *N,* виконуваних при

розвязку однієї задачі, причому значення *θ1, ..., θN* включають всі операції, зокрема що відносяться до системних процесів.

Відмінності в оцінках λ, одержуваних для різних індексів *n,* свідчить про погрішності у вимірюванні *Pn* і *θn*. Таким чином, оцінки продуктивності λ і *P1, ..., PN* однозначно зв'язані через характеристики *θ1, ..., θN* задач.

Хай відомі характеристики задач *θ1, ..., θN* і сумарна швидкодія *V1, ..., VN* пристроїв, що реалізовують операції типу 1, ..., *N* відповідно. У припущенні, що всі пристрої можуть працювати паралельно в часі і режим обробки задач, що задається управляючими програмами операційної системи, забезпечує паралельну роботу пристроїв, можна одержати верхню оцінку максимальної продуктивності системи

$$\lambda^* \; ? \; \min (V1/\theta 1, \ldots, VN/\theta N). \qquad (4.3)$$

Значення *Vn* /in визначають максимальну продуктивність пристроїв типу *n=1, ..., N,* що характеризується числом задач, які здатні обслужити пристрої за одиницю часу. Якнайменше продуктивна в заданому класі задач група однотипних пристроїв і визначить продуктивність системи. Оцінка (4.3) буде ближчим до реального значення λ*, якщо замість *V1, ..., Vn* підставити значення *V1*, ..., VN*,* характеризуючі комплексну продуктивність системи.

Час відповіді. *Час відповіді,* інакше час перебування завдань (задач) в системі, - тривалість проміжку часу від моменту надходження завдання в систему до моменту закінчення його виконання. На Рис. 4.11 вказаний час відповіді *u1, u2*, … .для завдань *J1, J2,* … відповідно.

У загальному випадку час відповіді — випадкова величина, що обумовлене наступними чинниками:

1) впливом початкових даних на число операцій введення, обробки і виведення даних і непередбачуваністю значень початкових даних;

2) впливом складу суміші задач, що одночасно знаходяться в системі, і непередбачуваністю складу суміші через випадковість моменту надходження задач на обробку.

Час відповіді як випадкова величина якнайповніші характеризується функцією розподілу *P(u<x)* або функцією густини вірогідності *p(u).* Частіше всього час відповіді оцінюється середнім значенням, яке визначається як статистичне середнє випадкової величини *ui, i* =1, …, *n*, спостережуваної для задач *Ji.*

Час відповіді складається з двох складових: часу виконання задачі і часу очікування. *Час виконання* задачі за відсутності паралельних процесів рівний сумарній тривалості всіх етапів процесу — введення, звернення до зовнішньої пам'яті, процесорної обробки і виведення. Час виконання задачі залежить від складності обчислень $^{3}\imath,\ldots,\theta_n$ і швидкодії $Vi,\ldots,V_N$ пристроїв 1,..., *N:*

Час очікування — сума проміжків часу, протягом яких задача знаходилася в стані очікування необхідних ресурсів. Очікування виникає при мультипрограмній обробці, коли ресурс, необхідний задачі, зайнятий іншою задачею і перша задача не виконується, чекаючи звільнення ресурсу. Час очікування залежить в першу чергу від режиму обробки задач і інтенсивності вхідного потоку задач (завдань).

Таким чином, час відповіді залежить від тих же параметрів, що і продуктивність: структури і характеристик технічних засобів, режиму обробки і характеристик задач. Залежність середнього часу відповіді *U* від інтенсивності вхідного потоку задач Λ приведена на Рис. 4.12. При Λ > 0 час відповіді $U > {}^{3}$, де ^{3} визначається (4.7). Із збільшенням Λ середній час відповіді монотонно зростає і може приймати скільки завгодно великі значення, якщо інтенсивність вхідного потоку Λ перевищує продуктивність системи λ* в перебіг скільки завгодно великого періоду часу.

Середній час відповіді характеризує швидкість реакції системи на вхідні дії: завдання, запити абонентів і т.п. Якість системи тим вище, чим менше середній час відповіді.

Характеристики надійності. *Надійність* — властивість системи виконувати покладені на неї функції в заданих умовах функціонування із заданими показниками якості: достовірністю результатів, пропускною спроможністю, часом відповіді і ін. Працездатність системи або окремих її частин порушується через відмови апаратури — виходу з ладу елементів або з'єднань.

Найважливіша характеристика надійності — *інтенсивність відмов,* визначаюча середнє число відмов за одиницю часу, як правило, за одну годину. Інтенсивність відмов залежить від числа елементів і з'єднань, що становлять систему. Якщо будь-яка відмова носить катастрофічний характер, тобто приводить до порушення працездатності системи, то інтенсивність відмов в системі, де лЯ — інтенсивність відмов і-го елементу або з'єднання і n — число елементів і з'єднань в системі. Так, якщо $л_0 = 10^2$ ч, то в середньому за 100 ч відбувається одна відмова. Середній проміжок часу між двома суміжними відмовами називається *середнім напрацюванням на відмову і* рівний $T_0 = 1/ л_0$.

Так, якщо $л_0 = 10^2$ год, то напрацювання на відмову складає 100 год. Проміжки часу між відмовами — випадкові величини з середнім значенням T_0, які, як правило, розподілені по експоненціальному закону. При цьому вірогідність того, що за час t відбудеться відмова, $P(t<x)=I-e-t/T_0$.

Працездатність системи, порушена в результаті відмови, відновлюється шляхом ремонту системи. Ремонт полягає у виявленні причини порушення працездатності — діагностиці системи і у відновленні працездатності шляхом заміни несправного елементу. Проміжок часу, що витрачається на відновлення працездатності системи, називається *часом відновлення.* Його тривалість залежить від складності системи, ступеня досконалості засобів діагностики і рівня ремонтопрігодності системи. Час відновлення — випадкова величина, що характеризується середнім значенням *ТБ — середнім часом відновлення.*

З урахуванням середнього напрацювання на відмову T_0 і середнього часу відновлення *Тв* надійність системи характеризується *коефіцієнтом готовності*

$$Кг = T_0 /(T_0 + ТБ), \qquad (4.4)$$

визначаючим частку часу, протягом якого система працездатна.

Значення 1-кг є часткою часу, протягом якого система непрацездатна, ремонтується. Так, якщо *Кг* = 0,95, то 95% часу система працездатна і 5% часу витрачається на її ремонт. Крім того, коефіцієнт готовності визначає вірогідність того, що в довільний момент часу система працездатна, а значення 1-кг — вірогідність того, що у цей момент часу система знаходиться в стані, відновлення.

Надійність системи може бути підвищена за рахунок резервування її елементів — дублювання, троїрованія і т.д. Проте резервування приводить до істотного збільшення вартості системи.

Вартість. *Вартість СОД* — це сумарна вартість технічних засобів і програмного забезпечення. Вартість технічних засобів визначається їх складом і технічними характеристиками. Пристрої з вищими технічними характеристиками — швидкодією, місткістю, надійністю — мають вищу вартість. Вартість програмного забезпечення визначається в основному витратами на розробку програм і тіражіруємостью програм — числом систем, в яких використовуються програми. Витрати на розробку програм найістотніше залежать від складності програм.

4.4. Режими обробки даних

Режим обробки даних — спосіб виконання завдань (задач), що характеризується порядком розподілу ресурсів системи між завданнями (задачами). Необхідний режим обробки даних забезпечується управляючими програмами операційної системи, які виділяють завданням оперативну і зовнішню пам'ять, пристрої введення — виведення, процесорний час

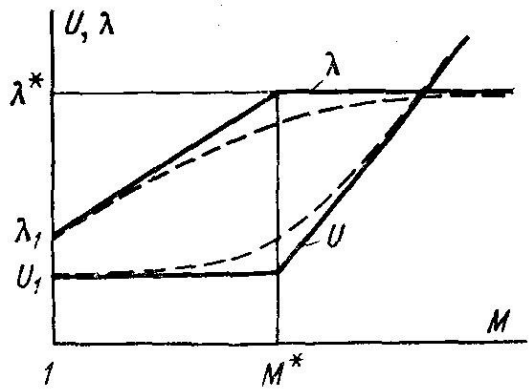
і інші ресурси у відповідному порядку з урахуванням атрибутів завдань — імен користувачів, пріоритетів завдань, складності задач і обчислень і ін.

Режим обробки даних породжує відповідний *режим функціонування системи,* що виявляється у порядку ініціації задач і представленні одним задачам переважного права на використовування ресурсів, в організації введення даних, зберігання програм в оперативній пам'яті, виведення даних і т.д. Порядок розподілу ресурсів між завданнями впливає на час перебування завдань в системі, продуктивність системи, вартість рішення задач і інші характеристики системи і процесів обробки задач. Вибір того або іншого режиму обробки даних обумовлений необхідністю забезпечення необхідних характеристик системи і процесів обробки. У свою чергу характеристики системи впливають на способи взаємодії користувачів з *системою*, а отже, на інтенсивність взаємодії, тривалість взаємодії і т.д. Таким чином, режим обробки даних пов'язаний з організацією процесу функціонування системи і відображається в першу чергу на характеристиках системи.

Розглянемо основні режими обробки даних і їх вплив на характеристики СОД.

Мультипрограмна обробка. У загальному випадку процес рішення задачі зводиться до послідовності етапів процесорної обробки, введення і виведення даних і звернень до зовнішніх пристроїв, що запам'ятовують. При цьому задача в кожен момент часу обробляється, як правило, одним пристроєм, а інші не можуть використовуватися до завершення роботи цього пристрою і, отже, можуть розподілятися для виконання інших задач. Режим обробки, при якому в системі одночасно обробляється декілька задач, називається *мультипрограмною обробкою* або, стисло, *мультипрограмуванням.* При цьому процеси обробки, що відносяться до різних задач, одночасно виконуються різними пристроями системи, здатними функціонувати паралельно. В цьому випадку говорять, що система обробки даних функціонує в *мультипрограмному режимі*. Мета мультипрограмування — збільшення продуктивності системи.

Число задач, що знаходяться в системі, називається *рівнем мультипрограмування.* Рівень мультипрограмування впливає на продуктивність і час відповіді системи таким чином. На рис.4.13 штриховими лініями показана залежність продуктивності системи λ і середнього часу відповіді *U* від рівня мультипрограмування *M. Прі* вивченні цих залежностей зручно криві *λ(M)* і *U(M)* представляти ламаними лініями, що складаються з двох прямих— асимптот і характеризують верхню і нижню оцінку: *λ(M)* і *U (M)* відповідно. У однопрограмному режимі (M=1) час відповіді *U = U1 = ·θ·*, де *³* визначається по формулі (4.7). При цьому продуктивність λ = λ1=1/U1.



Рисуное 4.13 – Вплив рівня мультипрограмування на продуктивність і час відповіді

Із збільшенням рівня мультипрограмування *M* збільшується вірогідність того, що більше число пристроїв одночасно зайняте виконанням задач. Але разом з тим вірогідність того, що декілька задач одночасно звертаються до одного пристрою, достатньо мала, і тому час очікування виявляється незначним. Проте при рівні мультипрограмування *M = M** виникає

ситуація, коли принаймні один пристрій виявляється повністю завантаженим. Подальше збільшення числа задач не приводить до зростання продуктивності λ, яка визначається продуктивністю λ* цього пристрою, але при *M>M** починає різко зростати час відповіді *U,* оскільки все більше число задач чекає моменту звільнення пристроїв'. Значення *M** називається *точкою насичення мультипрограмної суміші,* а також *точкою насичення системи* і залежить в першу чергу від числа пристроїв, які у складі системи можуть функціонувати паралельно. Чим більше число пристроїв, тим більше M*. Крім того, на значення *M** істотно впливають властивості задач. Якщо задачі переважно використовують один пристрій, то значення *M** невелике і може бути рівним одиниці. Якщо задачі завантажують всі пристрої, то значення *M** визначається числом пристроїв в системі.

Робота системи при рівні мультипрограмування *M>M** неефективна, оскільки немає виграшу в продуктивності і збільшується час відповіді.

Корисно зіставити характер зміни часу відповіді на рис.4.12 і 4.13. На рис.4.12 при інтенсивності вхідного потоку задач Λ>λ* час відповіді *U* > ?, а на рис.4.13 при рівні мультипрограмування M > M* час відповіді *U* > const, хоча в обох випадках продуктивність системи λ > λ*. Річ у тому, що при Λ>λ* система функціонує в нестаціонарному режимі, коли в одиницю часу поступає (Λ—λ*) завдань, які не можуть бути оброблені, якщо Λ>λ*. Тому теоретично число завдань в системі може опинитися скільки завгодно великим і в межі, при *t* > ?, — нескінченним. На рис.4.13 система завжди функціонує в стаціонарному режимі з постійним числом задач *M, що* знаходяться в ній. Тому час відповіді у всіх випадках кінцевий.

Продуктивність λ і середній час відповіді *Ó* зв'язані між собою залежністю

$$\lambda = M/U, \qquad (4.5)$$

яка називається *формулою Літтла* і є фундаментальним законом теорії масового обслуговування.

Число одночасно виконуваних задач, визначуване (4.15), називається *коефіцієнтом мультипрограмування* і рівне відношенню продуктивності системи в мультипрограмному режимі λ до продуктивності в однопрограмному режимі λ1 = 1/и, якщо витрати ресурсів на організацію мультипрограмування зростають пропорційно числу одночасно виконуваних задач, тобто

$$m = \lambda/\lambda 1. \qquad (4.6)$$

Таким чином, коефіцієнт мультипрограмування *m* є показником збільшення продуктивності системи за рахунок мультипрограмування. З (4.5) витікає, що коефіцієнт мультипрограмування

$$1 ? m ? N, \qquad (4.7)$$

де *N* — число пристроїв системи, здатних функціонувати паралельно з кожним з *N-1* решти пристроїв. При цьому передбачається, що система працює без відмов.

Оперативна і пакетна обробка даних. Стосовно СОД, призначеним для

інформаційного обслуговування користувачів (але не технічних об'єктів і систем), прийнято виділяти два режими обробки даних: оперативну і пакетну обробку. *Оперативна обробка даних* характеризується: 1) малим об'ємом вводяться — даних, що виводяться, і обчислень, що доводиться на одну взаємодію користувача з системою (на одну задачу); 2) високою інтенсивністю взаємодії і витікаючою звідси вимогою зменшення часу відповіді. Оперативна обробка необхідна в системах банківських, резервування квитків, довідкових і ін. *Пакетна обробка* даних характеризується: 1) великим об'ємом вводяться — даних, що виводяться, і обчислень, що доводиться на одну взаємодію користувача з системою (на одну задачу); 2) низькою інтенсивністю взаємодії і допустимістю великого часу відповіді. Пакетна обробка типова для обчислювальних центрів науково-технічного профілю, систем обробки обліковостатистичних даних, результатів геофізичних вимірювань і т.д.

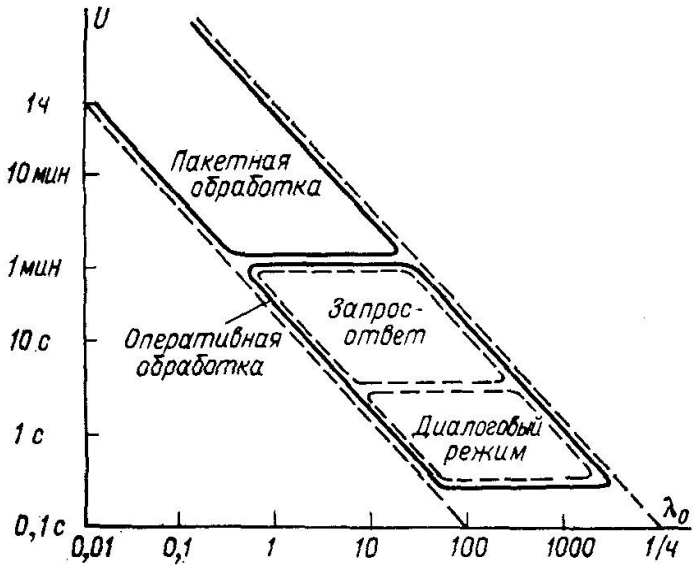


Рисунок 4.14 – Час відповіді та інтенсивність взаємодії користувачів з системою для різних режимів обробки

Області типових значень часу відповіді *U* і інтенсивності λ0 взаємодії користувача з системою, відповідні оперативній і пакетній обробці, представлені на рис.4.14. Твір ρ0 = λ0U визначає навантаження, створюване користувачем. Значення ρ0 можна розглядати, по-перше, як частку часу, протягом якого користувач обслуговується системою, і, по-друге, як вірогідність того, що в довільний момент часу система ініційована користувачем, тобто обслуговує його. Значення 1 – с0 визначає частку часу, протягом якого користувач не взаємодіє з системою, і одночасно вірогідність цього стану. Так, для пакетної обробки (л0=0,2 ч-1 ^{i} U = 0,15 ч) типове навантаження, створюване одним користувачем, с0 = 0,2·0,15=0,03.

В рамках оперативної обробки виділяють два режими: *запит — відповідь* і *діалоговий.* Режим запит – відповідь характеризується меншою інтенсивністю і більшою тривалістю взаємодії в порівнянні з діалоговим режимом. Типовий приклад використовування режиму запит—відповідь — довідкова служба на основі ЕОМ. При цьому користувач формує текст запиту, який вводиться в ЕОМ, і відповідь повинна бути одержаний за декілька десятків секунд. Робота в діалоговому режимі припускає практично миттєвий контакт користувача *з* системою, при якому система реагує на дії користувача із затримкою в декілька секунд або частки секунди. Найжорсткіші обмеження виникають, коли система повинна обслуговувати елементарні маніпуляції користувача, що працює за терміналом: наприклад, реагувати на натиснення кожної клавіші. В цьому випадку час відповіді не повинен перевищувати 0,1 з. Менш жорстким є режим, коли система повинна реагувати тільки на моменти закінчення набору рядків, забезпечуючи час відповіді, рівний декільком секундам. Швидкість реакції системи на дії користувача є неодмінною умовою діалогового режиму.

Діалоговий режим створює максимальні зручності для користувача, забезпечуючи постійний контроль даних (програм і початкових даних), що вводяться, мінімальний час відповіді, можливість оперативного втручання користувача в процес рішення задачі і оперативний доступ користувача до системи. За рахунок цього мінімізуються втрати через простій користувачів в очікуванні результатів, некоректних дій користувачів або «несподіваної» поведінки програм. Проте діалоговий режим забезпечується за рахунок використовування системи з великою продуктивністю, що вимагає великих капітальних вкладень. Крім того, вартість виконання програми в діалоговому режимі більше, ніж в пакетному, через чималі витрати, пов'язані з управлінням процесами з боку операційної системи.

При виконанні розрахунків, особливо складних, за апробованими програмами і методиками постійний контакт користувача з системою необхідний тільки на етапі введення даних. При їх обробці і виведення результатів оперативний контакт з користувачем не потрібен. У таких випадках найекономічнішим способом обробки даних є пакетний режим. У пакетному режимі організація процесу в системі має на меті не мінімізацію часу відповіді, а зниження вартості обробки даних за рахунок ефективного використовування ресурсів системи. У пакетному режимі управління процесами — вибір завдань з черги на обробку і порядок виконання задач — направлено на підвищення продуктивності системи за рахунок формування суміші задач, забезпечуючої максимальне завантаження по можливості всіх ресурсів системи. В цьому випадку час відповіді стає вельми значним: десятки хвилин і годинник.

Обчислювальні системи і комплекси в переважній більшості випадків (а мережі завжди) використовуються в *режимі колективного доступу,* при якому з системою взаємодіє колектив користувачів. Виконання завдань виробляється в мультипрограмному режимі. Забезпечення режимів пакетного, запит—відповідь і діалогового виробляється за рахунок відповідних способів управління ресурсами і процесами, реалізовуваних управляючими програмами операційної системи.

Обробка в реальному масштабі часу. У системах управління реальними об'єктами, побудованих на основі ЕОМ, процес управління зводиться до рішення фіксованого набору задач {*A1*, …, *AM*}. Кожна задача ініціюється або періодично, або при виникненні певних ситуацій в системі. При цьому темп ініціації задач і час отримання результатів обчислень жорстко регламентуються динамічними властивостями керованого об'єкту: технологічної установки, рухомого об'єкту і ін. Це означає, що на час рішення задач управління накладаються обмеження *u1* ? *U1**, …, *uM* ? *UM**, визначаючі граничний допустимий час відповіді *U*1, ..., U*M* для задач *A1*, ..., *AM* відповідно. Режим, при якому організація обробки даних підкоряється темпу процесів поза СОДОЮ, називається обробкою в *реальному масштабі часу* (РМВ).

Обробка в РМВ забезпечується за рахунок:

1) вибору структури СОД і швидкодії пристроїв відповідно до задач обробки А і вимогами до часу обробки *U*1, ..., U*M',*

2) способів організації процесів обробки, забезпечуючих необхідний час відповіді *u1* ? *U1**, …, *uM* ? *UM** при обмеженій продуктивності пристроїв і заданій структурі СОД.

Режим телеобробки даних. *Телеобробка* (видалена обробка) — режим обробки даних при взаємодії користувачів з СОД через лінії зв'язку. Телеобробка розглядається як самостійний режим обробки даних з наступних причин. По-перше, віддаленість користувачів від СОД і наявність між ними специфічного засобу передачі даних — лінії зв'язку — породжує необхідність в спеціальних діях користувачів при організації доступу до системи і завершенні сеансу роботи. По-друге, наявність ліній зв'язку накладає обмеження на форму і час обміну даними між користувачами і СОД. Ці обмеження приводять до необхідності спеціальних способів організації даних і доступу до них, що в свою чергу відображається на структурі прикладних програм, використовуваних в режимі телеобробки.

Режим телеобробки характеризується, перш за все, специфікою доступу користувача до системи і системи до даних, передаваних через видалені термінали, тобто пов'язаний в першу чергу з організацією обробки даних усередині СОД. При цьому користувачі можуть працювати в режимах пакетному, діалоговому або «запит—відповідь». Кожний з цих режимів характеризується специфічним способом взаємодії користувачів з системою і відповідним часом відповіді.

5. ОЦІНКА ТРУДОМІСТКОСТІ АЛГОРИТМУ.

5.1. Трудомісткість алгоритму

Задача, що підлягає вирішенню на РС, може бути охарактеризована кількістю даних, складністю алгоритму та його трудомісткістю. Під трудомісткістю алгоритму розуміється кількість обчислювальної роботи, необхідної для його реалізації. Трудомісткість характеризує витрати часу для реалізації алгоритму на деякій сукупності технічних засобів. Звичайно, трудомісткість оцінюється кількістю процесорних операцій та операцій введення-виведення. В загальному випадку, трудомісткість алгоритму є випадковою величиною, що залежить від вхідних даних. Тому, трудомісткість алгоритму може бути визначена тільки наближено, в термінах теорії ймовірності: математичними сподіваннями, дисперсією і т. д.

Трудомісткість алгоритму в першому наближенні може бути охарактеризована набором параметрів:

- середня кількість процесорних операцій, необхідних для однієї реалізації алгоритму;
- $N_1, N_2,\ldots,N_H$ - середня кількість запитів до файлів за один прогін програми; $L_1, L_2,\ldots,L_H$ - середня кількість інформації, що передається за одне звернення до файлів $F_1, F_2,\ldots,F_H$.

При необхідності набір параметрів, що характеризують трудомісткість алгоритму може бути доповнений. Вхідна інформація для розрахунку трудомісткості алгоритму може бути одержана з блок-схеми алгоритму (рис. 5.1).

Для розрахунку трудомісткості алгоритму необхідно знати ймовірності переходів з логічних вершин при одиничному значенні логічної умови. Якщо відповідну ймовірність визначити через *p*, тоді ймовірність виходу з логічної вершини при нульовому логічному значенні умови, що перевіряється буде дорівнювати *1 - p*. Для подальших розрахунків схему алгоритму раціонально зображати в вигляді графа алгоритму. Для цього пропонується перенумерувати всі оператори схеми алгоритму. У логічних операторів замість логічних умов '1' і '0' будемо записувати відповідну даному виходу ймовірність. Ймовірність виходу з операторної вершини дорівнює 1. Отриманий таким чином граф алгоритму зображений на рис. 2. Граф алгоритму можна істотно спростити, якщо трудомісткість виконання логічних вершин незначна в порівнянні з трудомісткістю виконання операторних вершин. Тоді стани, що відповідають логічним вершинам, можна злити з станами, що випереджають відповідні операторні вершини. В нашому прикладі можна злити стан (1.2), (3.4), (7.8), (10.11).

Після перенумерації отримаємо мінімізований граф алгоритму (рис. 3). При достатньому досвіді до нього можна було прийти відразу від блок-схеми (рис.5.1). Позначимо через $n_1, n_2,\ldots,n_{k-1}$ - середнє число запитів до операторів $V_1, V_2,\ldots,V_{k-1}$, відповідно. Кожен з операторів $V_i(i=1,2,\ldots,k-1)$ буде характеризуватись наступним набором чисел:

- k_i - середня кількість процесорних операцій, що виконуються в операторі V_i;
- m_{ij} - середня кількість запитів до файлу F_j(j=1,2,…,h) з оператора V_i;
- l_{ij} - середня кількість інформації, що передається при одному запиті до файлу $F_j(j=1,2,\ldots,H)$ з оператора V_i .

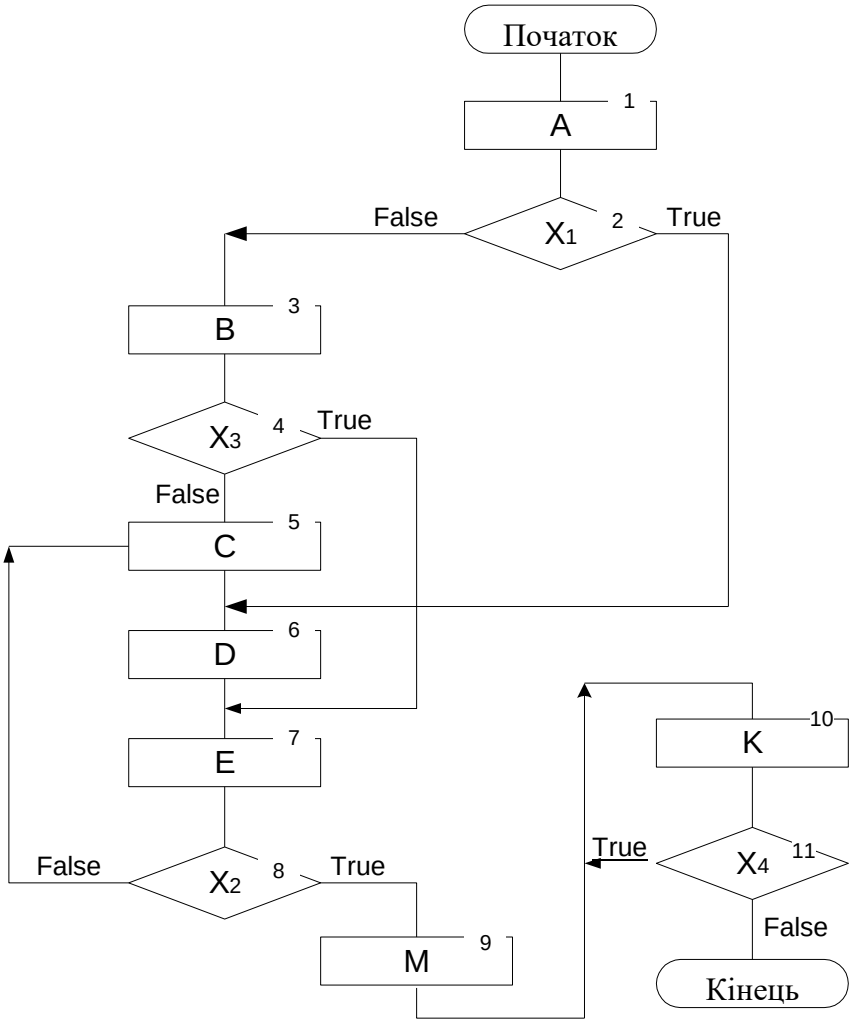


Рисунок 5.1 – Блок-схема алгоритму

Тоді для оцінки трудомісткість можна використати наступні формули:

$$\Theta = \sum_{i=1}^{k-1} n_i k_i ,$$

$$N_j = \sum_{i=1}^{k-1} n_i m_{ij} ,$$

$$L_j = \frac{1}{N_j} \sum_{i=1}^{k-1} n_i l_{ij} , \ (j=1,2,...,H)$$

де Θ - середнє число процесорних операцій, що виконуються при одному виконанні алгоритму; N_j- середнє число запитів до файлу $F_j(j=1,2,...,H)$ при одному виконанні алгоритму l_{ij} - середня кількість інформації, що передається при одному запиті до файлу $F_j(j=1,2,...,H)$ при одному виконанні програми.

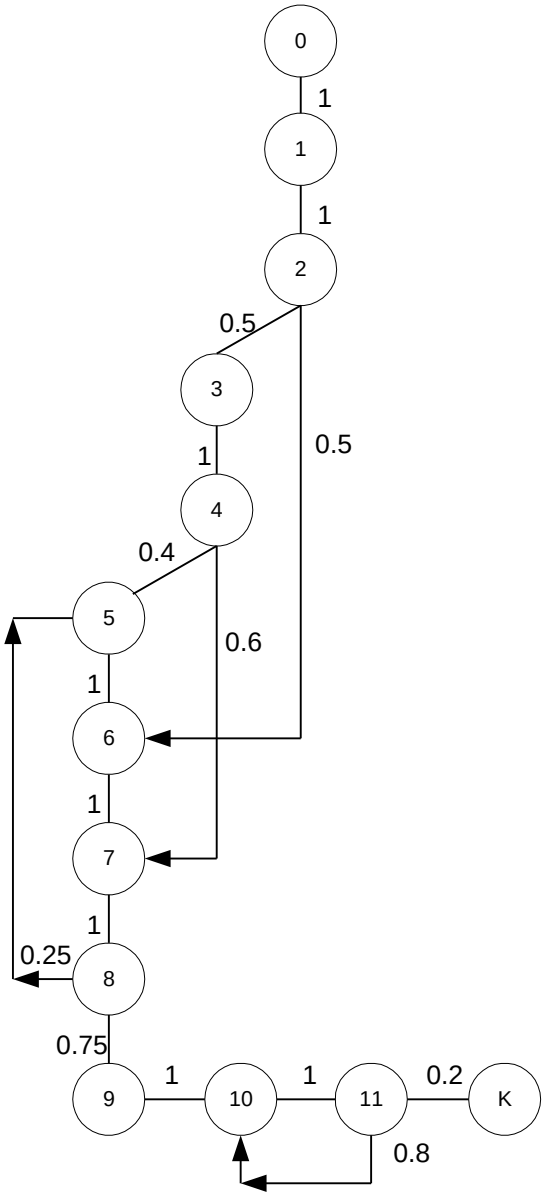


Рисунок 5.2 – Граф алгоритму

Для визначення середнього числа запитів n_i до оператора $V_i(i=1,2,...,k-1)$ звичайно робляться наступні допущення:

- ймовірність виконання оператора V_i після оператора V_j дорівнює P_{ij} і є постійною величиною;
- ймовірність P_{ij} залежить тільки від оператора V_i;

$$\sum_{i=1}^{k-1} P_{ij} = 1$$

При виконанні вищезазначених допущень обчислювальний процес стає Марківським процесом з станами $S_1,S_2,...,S_k$. При цьому оператори $V_1,V_2,...,V_{k-1}$ відповідають станам $S_1,S_2,...,S_{k-1}$. Стан S_k відповідає кінцевій вершині граф алгоритму. Стани $S_1,S_2,...,S_{k-1}$ називаються незворотними, бо процес, неодмінно, з них виходить.

Сучасна теорія алгоритмів налічує декілька способів оцінки трудомісткості алгоритмів.

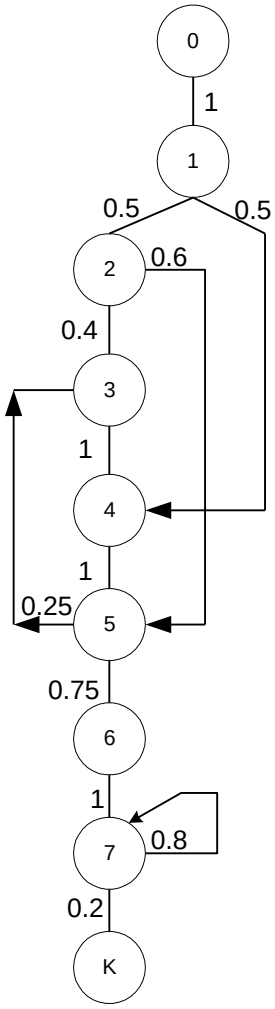


Рисунок 5.3 – Мінімізований граф алгоритму

5.2.Оцінка трудомісткості алгоритмів засобами теорії марківських ланцюгів.

Для визначення середнього числа процесорних операцій, що виконуються за один прогін програми необхідно подати граф алгоритму у вигляді стохастичної матриці *P*. Для рис.5.3 вона матиме вигляд:

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_k
S_0	1	0	0	0	0	0	0	0
S_1	0	0.5	0	0.5	0	0	0	0
S_2	0	0	0.4	0	0.6	0	0	0
S_3	0	0	0	1	0	0	0	0
S_4	0	0	0	0	1	0	0	0
S_5	0	0	0.25	0	0	0.75	0	0
S_6	0	0	0	0	0	0	1	0
S_7	0	0	0	0	0	0	0.8	0.2

Елементами матриці *P* є ймовірності переходу з стану і в стан j, що наведені на графі алгоритму. З стохастичної матриці слідує, наприклад, що в стані S_4 процес може виявитися при переході з S_1 з ймовірністю 0.5 або при переході з стану S_3 з ймовірністю 1.

Якщо відомі середні числа запитів до вершин V_1 і V_3 , то число запитів до вершини V_4 буде рівним: $n_4 = p_{14} n_1 + p_{34} n_3 = 0.9n_1 + n_3$. В загальному випадку можна записати (складання по стовпчику стохастичної матриці):

$$n_i = \sum_{j=0}^{k-1} p_{ji} n_j \quad , \quad (n_0 = 1, i = 1,2,...,k\text{-}1) \tag{4}$$

Останній запис являє собою систему лінійних алгебраїчних рівнянь, рішення яких дає середнє число запитів до операторів. Для прикладу з стохастичної матриці маємо:

$n_1 = 1n_0 = 1*1 = 1$; $n_2 = 0.5n_1 = 0.5*1 = 0.5$; $n_3 = 0.4n_2 + 0.25n_5 = 0.2 + 0.25n_5$; $n_4 = 0.5n_1 + n_3 = 0.5 + n_3$; $n_5 = 0.6n_2 + n_4 = 0.3 + n_4$; $n_6 = 0.75n_5$; $n_7 = n_6 + 0.8n_7$.

Вирішуючи систему рівнянь, отримаємо:

$n_1 = 1$; $n_2 = 0.5$; $n_3 = 0.533$; $n_4 = 1.033$; $n_5 = 1.333$; $n_6 = 1$; $n_7 = 5$.

Для перевірки правильності вирішення системи лінійних рівнянь можна використати тотожність:

$$n_k = \sum_{j=0}^{k-1} p_{jk} n_j = 1, \quad \text{при} \quad n_0 = 1 \tag{5}$$

В наведеному прикладі це виконується так: $n_k = 0.2 * n_7 = 0.2 * 5 = 1$.

Даний спосіб є універсальним (в випадку марківського процесу), але вимагає великих затрат часу при значних розмірах стохастичної матриці.

5.3. Мережевий підхід до оцінки трудомісткості алгоритмів

Мережевий підхід зручний для аналізу трудомісткості алгоритму вручну, та дозволяє обчислювати середню, мінімальну і максимальну трудомісткість алгоритму на графах, не що містять цикли. Для прикладу розглянемо граф алгоритму, зображений на рис. 5.4.

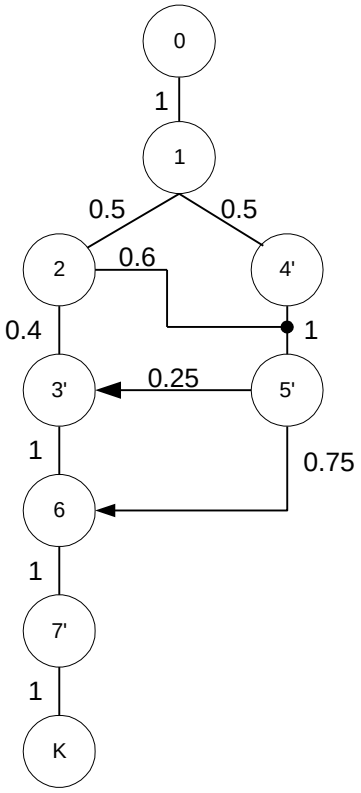


Рисунок 5.4 – Граф алгоритму без циклів

Використовуючи формулу (4) для рис.5.4 можемо записати:

$n_0 = 1$; $n_1 = 1 * n_0 = 1$; $n_2 = 0.5 * n_1 = 0.5$; $n_{4'} = 0.5 * n_1 = 0.5$;

$n_{5'} = n_{4'} + 0.6n_2 = 0.5 + 0.6 * 0.2 = 0.8$; $n_{3'} = 0.4n_2 + 0.25n_{5'} = 0.4 * 0.5 + 0.25 * 0.8 = 0.4$; $n_6 = n_{3'} + 0.75n_{5'} = 0.4 + 0.75 * 0.8 = 1$; $n_{7'} = 1 * n_6 = 1$.

В зв'язку з відсутністю циклів систему легко вирішувати. Витрати часу на аналіз можуть бути додатково знижені, якщо ввести ефективну нумерацію станів графа алгоритму. Остання полягає в тому, що номер вершини повинен бути таким, щоб дуги що входять в цю вершину починалися в вершинах з меншими номерами. Якщо після такої нумерації рівняння записувати в порядку зростання номерів вершин, то вони будуть відразу вирішуватися, бо невідомі з меншими номерами вже будуть обчислені. В наведеному прикладі нумерація вершин не відповідає визначеній вимозі, але рівняння записувалися в порядку їхнього вирішення. За наявності в графі алгоритму циклів, їх слід замінити операторами з еквівалентною трудомісткістю. Якщо в графі алгоритму є вкладені в цикл цикли, тоді, перш за все слід замінити еквівалентними операторами внутрішні цикли, а після цього переходити до заміни зовнішніх циклів. Розрахунок трудомісткості алгоритму виконується по наведеній вище методиці.

Визначимо через p_c ймовірність замикання циклу, тобто ймовірність переходу по дузі з кінця циклу в його початок. Тоді в відповідності з виразом (4) можна записати:

$$n_c = 1 + p_c n_c \,, \qquad (6)$$

де n_c - середнє число повторень циклу.

Тоді середнє число процесорних операцій циклу буде рівне: $k_c = n_c k_{mc}$, де k_{mc} - середня трудомісткість тіла циклу. Середнє число запитів до файлів і середня кількість інформації, що передається при зверненні в циклі до файлів, визначається аналогічно.

Приклад 1

В алгоритмі на рис. 3 є два цикли. Перший містить оператори V_3,V_4,V_5, а другий складається тільки з оператора V_7. Перший цикл ускладнюється входами в середину циклу з операторів V_1 та V_2 . Для позбавлення від входів в цикл не через початок циклу V_3 обчислимо

елементарні перетворення графа алгоритму. Еквівалентний граф після очевидних перетворень зображений на рис.5.5.

З рис.5.5 слідує, що оператори V_4 та V_5, що використовувались для входу в тіло циклу не через його початок, винесені окремо під номерами 4' й 5'.

Кількість повторень циклів з виразу (7) дорівнює:

$$n_{c1} = \frac{1}{1-0.25} = 4;$$

$$n_{c2} = \frac{1}{1-0.8} = 5;$$

де n_{c1}, n_{c2} - число повторень першого та другого циклів. Граф алгоритму без циклів наведений на рис.5.5

Наведений вище підхід дозволяє обчислити середню трудомісткість алгоритму. Для оцінки максимальної і мінімальної трудомісткості алгоритму необхідно перебрати всі можливі шляхи, ведучі з початкової вершини графа алгоритму в кінцеву, і вибрати з них шляхи, що дадуть максимальну і мінімальну трудомісткість. Слід врахувати, що, число запитів до файлів для шляху з мінімальною кількістю процесорних операцій не обов'язково є мінімальним. За наявності циклів в графі алгоритму, для тіла циклу визначається мінімальна і максимальна трудомісткість. Знаходиться мінімальне і максимальне число повторень циклу і формуються відповідні еквівалентні по трудомісткості оператори.

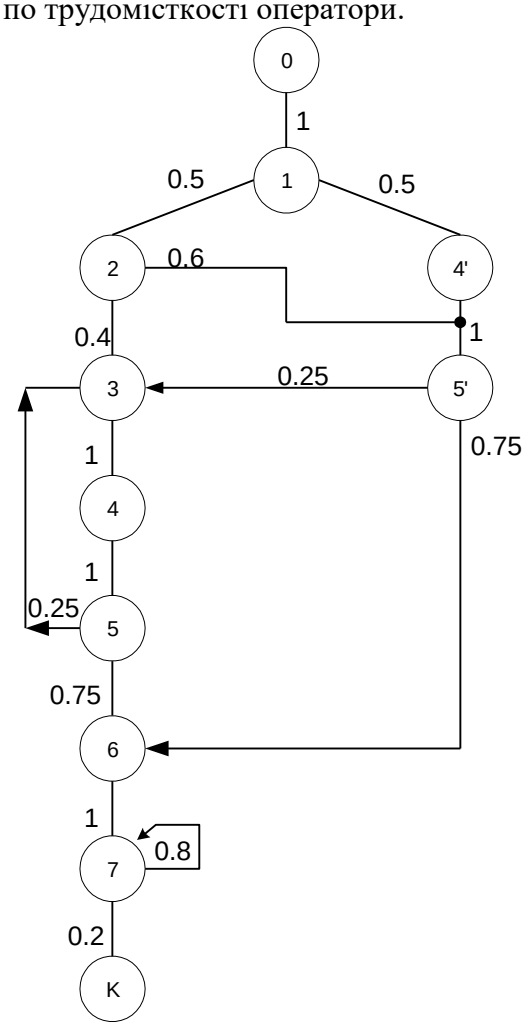


Рисунок 5.5 – Еквівалентний граф алгоритму

Приклад 2

Оцінимо мінімальне і максимальне число процесорних операцій для графа алгоритму, зображеного на рис. 5.4. Для спрощення приймемо, що всі оператори мають трудомісткість, рівну 1000 процесорних операцій. Через Ai і Bi будемо визначати, відповідно, мінімальне і максимальне число процесорних операцій, що буде мати місце в момент виходу процесу з i-ї вершини графа.

$A_0 = 0;$

$A_1 = min(A_0) + 1000 = 1000;$

$A_2 = min(A_1) + 1000 = 2000;$

$A_{4'} = min(A_1) + 1000 = 2000;$

$A_{5'} = min(A_2,A_{4'}) + 1000 = 3000;$

$A_{3'} = min(A_2,A_{5'}) + 1000 = 3000;$

$A_6 = min(A_{3'},A_{5'}) + 1000 = 4000;$

$A_7 = min(A_6) + 1000 = 5000;$

$B_0 = 0;$

$B_1 = max(B_0) + 1000 = 1000;$

$B_2 = max(B_1) + 1000 = 2000;$

$B_{4'} = max(B_1) + 1000 = 2000;$

$B_{5'} = max(B_2,B_{4'}) + 1000 = 3000;$

$B_{3'} = max(B_2,B_{5'}) + 1000 = 4000;$

$B_6 = max(B_{3'},B_{5'}) + 1000 = 5000;$ $B_7 = max(B_6) + 1000 = 6000.$

6. ВИЗНАЧЕННЯ ШВИДКОДІЇ ЕОМ.

До найважливіших характеристики ЕОМ відносять:

- Продуктивність ЕОМ;
- Ємність ОЗП і ЗЗУ;
- Система команд;
- Програмне забезпечення; • Надійність; • Вартість.

Вище наведені характеристики зв'язані між собою складними функціональними залежностями, що визначаються засобами організації, елементною базою, технологією виробництва і т. п. Характеристики ЕОМ в більшості випадків є векторними величинами.

Продуктивність ЕОМ - характеристика обчислювальної потужності, та визначає кількість обчислювальної роботи, що виконується на ЕОМ за одиницю часу. Дане визначення продуктивності не є конструктивним, тому що відсутнє чітке, інтуїтивно зрозуміле для користувача, поняття “одиниці обчислювальної роботи”. По цій причині на практиці користуються різноманітними наборами чисел, що характеризують продуктивність ЕОМ.

Номінальна продуктивність - вектор $V=(V1, V2,..., Vn)$, складатися з швидкодії пристрої, що входять в ЕОМ. Наприклад, швидкодія процесора, оперативної пам'яті, каналів, накопичувачів, пристроїв введення-виведення. Під швидкодією розуміється кількість операцій, що виконуються за одиницю часу. Поняття “номінальної продуктивності” зручне для виробників ЕОМ і характеризує потенційні можливості ЕОМ в сенсі швидкості обробки даних.

Наявність загальних ресурсів в ЕОМ не дозволяє повністю використати їх номінальну продуктивність. Наприклад, наявність в структурі одного селекторного каналу не дозволяє читати (або записувати) інформацію з двох (чи більше) накопичувачів на магнітних дисках, одночасно. Тому інколи вводять поняття комплексної продуктивності ЕОМ, що характеризує не тільки швидкодію пристроїв, але і структуру зв'язків між ними, можливість паралельної роботи частини пристроїв.

Складність структури сучасних ЕОМ призвела до необхідності створення і постійної експлуатації комплексу програм, що здійснюють управління обчислювальним процесом і реалізують найбільш загальні алгоритми обробки інформації. Цей комплекс програм відомий під назвою “операційна система”. Структура технічних засобів ЕОМ і порядок проходження задачі користувача через ЕОМ, що визначається операційною системою, характеризує системну продуктивність ЕОМ.

Врахування параметрів задач користувача (трудомісткість алгоритму, кількість звернень до пристроїв введення-виведення, необхідна ємність ОП та інше) призводять визначення поняття продуктивності з точки зору навантаження користувача (користувацька продуктивність). Така, “користувацька продуктивність” характеризується кількістю задач користувача, що вирішуються за одиницю часу. Користувацька продуктивність важлива для користувача, але мало актуальна для виробників універсальних ЕОМ.

Відомо, що найважливішим параметром продуктивності ЕОМ є його швидкодія, іншими словами, швидкість обробки інформації. Найчастіше під швидкодією ЕОМ розуміють швидкість або час виконання тільки процесорних операцій.

Швидкодію ЕОМ визначають: швидкодія процесора та час звернення до ОЗП. В загальному випадку швидкодія ЕОМ істотно відрізняється для різних процесорних операцій. Процесорні операції, в свою чергу, характеризуються числом звернень до ОЗП або регістрів загального призначення, алгоритмами обробки, вхідними даними. Тому для характеристики швидкодії ЕОМ можна ввести поняття номінальної швидкодії ЕОМ, що визначається вектором значень $V1, V2,..., V_M$, де M - число операцій, що виконуються ЕОМ, $Vi\ (i=1.2,..., M)$ - середнє число операцій *i-го* типу, що виконуються за секунду, t_i (i=1.2,..., M) - середній час виконання *i-ї* операції.

Порівняння ЕОМ по номінальній швидкодії є дуже важкою задачею важко із-за великого числа операцій та їхнього різноманітного складу. Але в свою чергу величина ймовірності виконання операції p_i визначається задачею, що вирішується. В нинішній час для порівняння різноманітних ЕОМ використовують, так звані суміші Гібсона, що задають ймовірність використання операцій для різноманітних класів задач, що вирішуються.

Практичні значення *Vi* або t_i (i=1.2,..., M) для конкретної ЕОМ, при різних значеннях навантаження можуть бути визначені декількома методами:

1) Використати відоме значення *i-ї* операції алгоритму в ЕОМ. На основі технічної документації складається граф-схема мікропрограми заданої операції і для конкретних вхідних даних визначається число тактових імпульсів. Період тактових імпульсів вимірюється безпосередньо на ЕОМ (з допомогою осцилографа чи інших пристроїв).

2) В процесорі заблокувати нарощування лічильника адреси команд (якщо це передбачене конструкцією ЕОМ) та запустити ЕОМ на виконання заданої операції в автоматичному режимі. З допомогою осцилографа визначити період виконання операції. Слід відзначити, що даний метод може інколи дати невірний результат для тих команд, що застосовують значення одного чи декількох операндів в ході свого виконання.

3) Сучасні ЕОМ мають таймери - пристрій для виміру тимчасових інтервалів з достатньо високою точністю. Суть методу полягає в наступному: запускається на виконання деяка програма та оцінюється час її виконання по показах таймера. Потім запускається на виконання друга програма, що відрізняється від попередньою тільки наявністю додаткової операції. Заміряється та оцінюється час виконання цієї програми. Очевидно, що ризниця часів виконання першої та другої програми дозволяє однозначно встановити часове значення додаткової операції.

7. СИНТЕЗ СИСТЕМИ ОПЕРАТИВНОЇ ОБРОБКИ МІНІМАЛЬНОЇ КОНФІГУРАЦІЇ

7.1 Визначення параметрів середньої задачі

Система оперативної обробки (СОО) являє собою сукупність технічних засобів, зайнятих обслуговуванням п'яти задач, що надходять в систему в випадкові моменти часу з відомою інтенсивністю. Окрім означених п'яти задач в системі вирішуються фонові задачі, що мають самий низький пріоритет, і на час перебування п'яти задач в системі не впливають. Наявність фонових задач призводить до того, що вартість процесора, яка припадає на СОО, пропорційна необхідній швидкодії процесора, що забезпечується частиною продуктивності серійного процесора. В варіантах завдань, місткості запам`ятовуючих пристроїв також є несерійними, що пояснюється присутністю фонових задач. Вхідними даними для виконання лабораторної роботи є:

1. Характеристики задач:
2. Характеристики пристроїв:
3. Вартісні характеристики:

Синтез СОО мінімальної конфігурації базується на тому, що коефіцієнт завантаження кожної з систем масового обслуговування, що входить в розімкнуту стохастичну мережу масового обслуговування, яка є математичною моделлю проектованої СОО, не перевищує одиниці. Це призводить до того, що в СОО, що синтезується, не може виникнути нескінченна черга, тому в такій системі можливо існування стаціонарного режиму.

Синтез СОО мінімальної конфігурації включає ряд етапів:

- Визначення характеристик середньої задачі;
- Визначення мінімальної швидкодії процесора;
- Визначення можливості розміщення файлів на різноманітних зовнішніх запам`ятовуючих пристроях;
- Визначення кількості зовнішніх запам`ятовуючих пристроях;
- Розподіл файлів на зовнішніх запам`ятовуючих пристроях;
- Визначення кількості селекторних каналів;
- Розподіл зовнішніх запам`ятовуючих пристроїв між селекторними каналами;
- Розрахунок часу перебування задач в СОО мінімальній конфігурації;
- Розрахунок вартості мінімальної конфігурації.

Мета даного етапу - отримати параметри середньої задачі однорідного потоку, що створює таке ж навантаження на обчислювальну систему, як і вхідний неоднорідний потік запитів на вирішення задач.

Параметри середньої задачі визначаються в такому порядку:

А) інтенсивність потоку запитів на вирішення середньої задачі

$$I_{no} = \sum_{i=1}^{n} I_n(i),$$

Б) середня трудомісткість в процесорних операціях при вирішенні середньої задачі

$$R_o = (\sum_{i=1}^{n} I_n(i) * R(i)) / I_{no},$$

В) середнє число запитів до файлу *F(j)*

$$D(j) = (\sum_{i=1}^{n} I_n(i) * N(i,j)) / I_{no}, \quad j=1, 2,..., 10,$$

Г) сумарне число запитів до файлів в процесі виконання середньої задачі

$$D_o = \sum_{j=1}^{10} D(j),$$

Д) ймовірність використання *j-го* файлу при вирішенні задач

$$P(j)=D(j)/(D_o+1),$$

Е) середня трудомісткість етапу розрахунків

$$R_p=R_o/(D_o+1),$$

Ж) ймовірність виходу задачі з системи

$$P_o= 1/(D_o+1).$$

7.2 Визначення мінімальної швидкодії процесора

Позначимо через V_{pr} швидкодію процесора. Тоді, мінімальна швидкодія процесора визначається по формулі:

$$V_{pr}(min)=1.1*I_{no}* R_o. \quad (8)$$

Коефіцієнт завантаження процесора Q_{pr} з врахуванням параметрів середньої задачі обчислюється по формулі:

$$Q_{pr}=I_{no}*R_o/ V_{pr}(min)< 1, \quad (9)$$

7.3 Визначення можливості розміщення файлів на зовнішніх запам`ятовуючих пристроях

На даному етапі слід визначити, на якому з двох типів зовнішніх запам`ятовуючих пристроїв розміщувати файли (вінчестер чи ГМД). Звичайно зберігання інформації на ГМД обходиться дешевше, але час доступу до цієї інформації значно перевищує час доступу до записів на вінчестер.

Послідовність розрахунків наступна:

А) інтенсивність звернення до *j-го* файлу

$$I_{nn}(j)=I_{no}*D(j), j=1, 2,..., 10, \quad (10)$$

Б) граничний час доступу до інформації

$$T(j)=1/I_{nn}(j), j=1, 2,..., 10. \quad (11)$$

Якщо $T(j)< T_{gml}$, то файл $F(j)$ слід розмістити на вінчестер. Якщо, $T(j)> T_{gml}$, то файл $F(j)$ можна розміщувати на ГМД.

7.4 Визначення кількості зовнішніх запам`ятовуючих пристроїв

Кількість зовнішніх запам`ятовуючих пристроїв знаходиться з двох умов: коефіцієнт завантаження менший одиниці та можливість розміщення файлів по місткості.

Розрахунок кількості зовнішніх запам`ятовуючих пристроїв проводиться в наступному порядку:

А) кількість звернень до вінчестера:

$$D_{nmd} = \sum D(j),\ j \in \text{вінчестеру},$$

Б) кількість звернень до ГМД:

$$D_{gml} = \sum D(j),\ \text{j} \in \text{ГМД},$$

В) інтенсивність запитів до вінчестера:

$$I_{nmd}=I_{no}* D_{nmd},$$

Г) інтенсивність звернення до ГМД:

$$I_{gml}=I_{no}* D_{gml},$$

Д) кількість вінчестерів по коефіцієнту завантаження:

$$Z_{1nmd} > I_{nmd}* T_{nmd},$$

Е) кількість ГМД по коефіцієнту завантаження

$$Z_{1gml} > I_{gml}* T_{gml},$$

Ж) кількість вінчестерів по місткості:

$$Z_{2nmd} \geq \sum \frac{M(j)}{M_{nmd}},\ j \in \text{вінчестеру},$$

З) кількість ГМД по місткості:

$$Z_{2gml} \geq \sum \frac{M(j)}{M_{gml}},\ j \in \text{ГМД},$$

И) вибір кількості вінчестерів і ГМД для проектованої системи:

$$Z_{nmd}=max(Z_{1nmd}, Z_{2nmd}),$$

$$Z_{nml}=max(Z_{1nml}, Z_{2nml}).$$

7.5 Розміщення файлів на зовнішніх запам‘ятовуючих пристроях

При розміщенні файлів на накопичувачах зовнішньої пам'яті потрібно керуватися наступними принципами:

А) файл, по можливості, повинен цілком міститься в накопичувачі;

Б) інтенсивність звернення до вінчестера та ГМД повинна бути приблизно однаковою;

В) коефіцієнт завантаження кожного накопичувача не повинен бути вищим одиниці;

Г) не можна перевищувати ємність накопичувача.

7.6 Визначення кількості селекторних каналів

Послідовність розрахунку кількості селекторних каналів включає в себе:

А) інтенсивність звернення до селекторних каналів:

$$I_{ck}=I_{no}*D_{o},$$

Б) ймовірність звернення до вінчестера:

$$P_{nmd} = \sum P(j),\ j \in \text{вінчестеру},$$

В) ймовірність звернення до ГМД:

$$P_{gml} = \sum P(j),\ j \in \text{ГМД},$$

Г) середня довжина запису у разі звертання до вінчестера:

$$L_{nmd} = \sum L(j)* P(j)/P_{nmd},\ j \in \text{вінчестеру},$$

Д) середня довжина запису у разі звертання до ГМД:

$$L_{gml} = \sum L(j)* P(j)/P_{gml},\ j \in \text{ГМД},$$

Е) середній час передачі середнього запису через селекторний канал:

$$T_{ck}=L_{nmd}*P_{nmd}/V_{nmd}+L_{nml}*P_{gml}/V_{gml},$$

Ж) кількість селекторних каналів:

$$Z_{ck} \geq I_{ck}*T_{ck}.$$

7.7 Розрахунок середнього часу відповіді та вартості СОО для мінімальної конфігурації

Середній час відповіді в СОО визначається виразом

$$U_{min}=U_{pr}+D_{nmd}*U_{nmd}+D_{nml}*U_{nml}+D_0*U_{ck},$$

де $U_{pr}=R_o/(V_{pr}(min) - I_{no}*R_o)$ - середній час перебування середньої заявки в процесорі;

$U_{nmd}=T_{nmd}/(1 - I_{nmd}*T_{nmd}/Z_{nmd})$ - середній час перебування на заявки на вінчестері;

$U_{gml}=T_{gml}/(1 – I_{gml}*T_{gml}/Z_{gml})$- середній час перебування заявки на ГМД;

$U_{ck}=1 - I_{ck}*T_{ck}/Z_{ck}$ - середній час перебування заявки в селекторному каналі.

Вартість СОО визначається по формулі:

$$S_{min}=D_p*V_{pr}(min)+Z_{ck}*S_{ck}+Z_{nmd}*S_{nmd}+Z_{nml}*S_{nml}.$$

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ларионов А.М., Майоров С.А., Новиков Г.И. Вычислительные комплексы, системы и сети. - Л.: Энергоатомиздат; Ленинградское отделение, 1987. - 288 с.
2. Основы теории вычислительных систем/ Под ред. С.А. Майорова. - М.: Высш. шк., 1978. - 408 с.
3. Голованов О.В., Дуванов С.Г., Смирнов В.Н. Моделирование сложных дискретных систем на ЭВМ третьего поколения. - М.: Энергия, 1978. - 160 с.
4. Коваль Г.И. и др. Программирование в системе виртуальных машин ЕС ЭВМ: Справочное издание / Г.И. Коваль, Т.М. Коротун, Е.М.Лавришева. - М.: Финансы и статистика, 1990. - 256 с.
5. Основы теории вычислительных систем / Под ред. С.А. Майорова. - М.: Высшая школа, 1978. - 408 стр.
6. Майоров С.А., Новиков Г.И. “Структура ЭВМ” - Л.: Машиностроение. Ленингр. отдние, 1979. - 384 с.
7. Дроздов Е.А., Комарницкий В.А., Пятибратов А.П. “ЭВМ ЕС” - М.: Машиностроение, 1981. - 648 с.
8. Каган Б.М. “ЭВМ и системы”- М.: Энергоатомиздат, 1985. -552 с.
9. Феррари Д. “Оценка производительности вычислительных систем” - М.: Мир, 1981.- 576 с.
10. Ларионов А.М., Майоров С.А., Новиков Г.И. Вычислительные комплексы, системы и сети. - Л.: Энергоатомиздат; Ленинградское отделение, 1987. - 288 с.
11. Голованов О.В., Дуванов С.Г., Смирнов В.Н. Моделирование сложных дискретных систем на ЭВМ третьего поколения. - М.: Энергия, 1978. - 160 с.
12. Коваль Г.И. и др. Программирование в системе виртуальных машин ЕС ЭВМ: Справочное издание / Г.И. Коваль, Т.М. Коротун, Е.М.Лавришева. - М.: Финансы и статистика, 1990. - 256 с.
13. Ахо А., Хопкрофт Д., Ульман Д. Структуры данных и алгоритмы. - М.: Издательский дом «Вильяме», 2001, - 384с.
14. Кормен Т., Лейзерсон Ч., Ривест Р. "Алгоритмы: постоение и анализ". - М.: МЦНМО 2000.
15. Кнут Д. "Искусство программирования" в 3-х томах - М.: Вильямс 2001.