ТЕМА 1. ОСНОВНІ ХАРАКТЕРИСТИКИ МІКРОПРОЦЕСОРІВ

Більш ніж чверть століття, починаючи з кінця Другої світової війни, ЕОМ залишалися громіздкими, коштовними та доступними лише вузькому колу фахівців пристроями, не дивлячись на те, що їх елементна база постійно вдосконалювалася – від електронних ламп до транзисторів, а від них – до інтегральних схем.

Ситуація змінилася докорінним чином у 1971 році, коли група фахівців корпорації Intel – Тед Хофф, Фредеріко Феджин та Стен Мейзор – створили перший мікропроцесор.

Історія творення першого в світі мікропроцесора почалася у 1969 році, коли Intel отримала від японської фірми Busicom замовлення на розробку першого набору мікросхем для родини калькуляторів, яка вироблялася цією фірмою. Початкова конструкція калькуляторів Busicom передбачала використання близько десяти мікросхем. Але Хоффу, Феджину та Мейзору вдалося спроектувати єдину універсальну мікросхему – центральнийпроцесор ЕОМ загального користування. Цей процесор узяв на себе виконання цілого комплексу різноманітних функцій, для реалізації яких раніше доводилося використовувати велику кількість різноманітних вузькоспеціалізованих компонентів.

Концепцію процесора загального призначення запропонував Тед Хофф. Ним же була розроблена архітектура універсальної мікросхеми, яка вміщувала в собі близько 2000 транзисторів. Система команд першого мікропроцесора була розроблена Стеном Мейзором, а саму мікросхему спроектував Федеріко Феджин. В середині листопада 1971 року Intel офіційно оголосила про створення універсального МП, який отримав фірмову назву «мікросхема 4004».

Про значення революції в обчислювальній техніці, яку викликала поява МП красномовно свідчить той факт, що у 1996 році прізвища Теда Хоффа, Фредеріко Феджина та Стена Мейзора були внесені до переліку лауреатів Національного залу Слави винахідників США, і тепер вони стоять поруч з іменами Томаса Едісона, Олександра Белла та інших видатних американських винахідників.

Саме поява МП відкрила шлях створенню персональних комп’ютерів, використанню цифрової обробки сигналів у радіотехніці та ін. МП є основою сучасної обчислювальної техніки – як універсальної, так і спеціалізованої.

Мікропроцесор є основним обчислювальним блоком комп'ютера. Це пристрій, що виконує програму - послідовність команд (інструкцій), написану програмістом і оформлену у вигляді модуля, що виконується. МП сімейства Intel x86, початок якому було покладено в 1979 році випуском моделі 8086, є основою побудови IBM PC- сумісних персональних комп'ютерів, хоча сфера їхнього застосування не обмежується ПК даної архітектури.

1. Класифікація і основні характеристики мікропроцесорів

Мікропроцесором у кібернетиці називають програмно-керований пристрій обробки інформації, виконаний на одній великій інтегральній мікросхемі (ВІС) або на певному наборі ВІС. С точки зору теорії цифрових пристроїв, МП –це найбільш складний на сьогоднішній день багатофункціональний цифровий автомат, який вміщує як послідовностні так і комбінаційні вузли.

1.1. Класифікація мікропроцесорів

1. *За функціональним призначенням розрізняють універсальні і спеціалізовані МП.*

Універсальні МП мають алгоритмічно універсальний набір команд, за допомогою якого можна здійснювати перетворення інформації відповідно до будь-якого заданого алгоритму. Продуктивність (швидкодія) таких процесорів практично не залежить від специфіки розв'язуваних задач.

Спеціалізовані МП призначені для рішення обмеженого і строго визначеного кола задач, іноді навіть для рішення однієї конкретної задачі. До спеціалізованих МП належать: сигнальні; медійні та мультимедійні; трансп’ютери; мікроконтролери.

Сигнальні процесори (процесори цифрових сигналів) призначені для цифрової обробки сигналів у реальному масштабі часу (наприклад, фільтрація сигналів, обчислення згортки та кореляційної функції, підсилення, обмеження та трансформація сигналу, пряме та обернене перетворення Фур’є).

Медійні та мультимедійні процесори призначені для обробки аудіо сигналів, графічної інформації, відеозображень, а також для розв’язування ряду задач у мультимедіакомп’ютерах, іграшкових приставках, побутової техніці.

Трансп’ютери призначені для масових паралельних обчислень і роботи у мультипроцесорних системах. Для них характерним є наявність внутрішньої пам’яті та вбудованого міжпроцесорного інтерфейсу, тобто каналів зв’язку з іншими МП.

Серед спеціалізованих МП також можна виділити *мікроконтролери* -- МП, призначені для рішення задач керування якими-небудь процесами або пристроями.

Класифікація МП за функціональним призначенням подана на рис. 1.

Мікропроцесори
Універсальні
Спеціалізовані
Сигнальні МП
Медійні та мультімедійні
Трансп’ютери
Мікроконтролери

Рис. 1.1. Класифікація МП за функціональним призначенням

На допомогу центральному процесорові в комп'ютер іноді вводять *співпроцесори*, орієнтовані на ефективне виконання яких-небудь специфічних функцій. Раніше широко були поширені математичні співпроцесори, що обробляють числові дані у форматі з плаваючою комою; *графічні співпроцесори*, що виконують геометричні побудови й обробку графічних зображень; *співпроцесори введення-виведення*, що розвантажують центральний процесор від нескладних, але численних операцій взаємодії з зовнішніми пристроями. Можливі й інші співпроцесори, однак усі вони несамостійні - виконання основного обчислювального процесу здійснюється центральним процесором, що відповідно до програми видає "завдання" співпроцесорам на виконання їхніх допоміжних функцій.

2. *За числом ВІС, використовуваних для побудови функціонально повного МП, розрізняють однокристальні, багатокристальні і багатокристальні секційні МП.*

Однокристальні МП реалізуються у вигляді один ВІС. Логічна структура багатокристального МП розбивається на складні функціональні вузли, кожен з яких реалізується у вигляді окремої ВІС.

Однокристальні МП можуть бути одноядерними та багатоядерними. *Багатоядерний процесор* має фактично декілька обчислювальних пристроїв, розташованих на одному кристалі, яки можуть працювати паралельно та мають загальні кола введення-виведення.

Багатокристальні секційні (розрядно-модульні) МП використовуються для побудови багаторозрядних МП на основі паралельно включенихмікропроцесорних секцій. Мікропроцесорна секція являє собою ВІС для обробки декількох розрядів даних (від 2 до 16), що може використовуватися як самостійно, так і як модуль для побудови МП, що обробляє більш довгі кодові слова.

Переважна більшість сучасних універсальних МП є *однокристальними*.

3. *За розрядністю оброблюваних кодових слів даних МП можуть бути з фіксованою або нарощуваною розрядністю слів.*

Процесорами з нарощуваною розрядністю кодів слів є тільки багатокристальні секційні МП, всі інші МП обробляють слова фіксованої розрядності.

4. *За способом керування обчислювальним процесом розрізняють МП із мікропрограмним і з схемним (апаратним) керуванням.*

Мікропроцесори зі схемним керуванням мають фіксований набір команд, розроблений фірмою-виробником, який не може змінювати користувач. У МП з мікропрограмним керуванням систему команд розробляють при проектуванні конкретного мікропроцесорного комплекту на базі наборунайпростіших мікрокоманд з урахуванням класу задач, для яких призначений МПК.

Взагалі *мікропроцесорним комплектом* називають сукупність інтегральних схем, сумісних за електричними, інформаційними та конструктивними параметрами і призначених для побудови електронно-обчислювальної апаратури та мікропроцесорних систем керування.

7. За типом архітектури, або принципом побудови розрізняють МП з нейманівською архітектурою та МП з гарвардською архітектурою.

8. За типом системи команд розрізняють CISC (Complete Instruction Set Computing) – процесори з повним набором команд, і RISC(Redused Instruction Set Computing) – процесори зі зменшеним набором команд.

1.2. Основні характеристики МП.

МП, будучи складним обчислювальним пристроєм, реалізованим у вигляді ВІС, характеризується багатьма параметрами і властивостями.

Як *ВІС*, МП може бути охарактеризований наступними параметрами: ступенем інтеграції (розмірами кристала і кількістю транзисторів на ньому), типом корпуса, числом виводів, кількістю і величинами живлячих напруга, потужністю що розсіюється, робочим температурним діапазоном, надійністю, навантажувальною здатністю і т.п. Ці параметри істотно розрізняються у різних процесорів, і мають суттєве значення лише для розроблювачів апаратного забезпечення ЕОМ.

Розроблювачі програмного забезпечення ЕОМ використовують інтегральні системні характеристики МП як обчислювального пристрою, до яких, наприклад, відносяться:

- використовувана система команд;
- структура системи переривань;
- можливість організації різних структур пам'яті і типів обміну інформацією з зовнішніми пристроями;
- способи організації спільної роботи декількох процесорів і ін.

Найбільше практично важливими для кінцевого користувача конструктивними і функціональними характеристиками є:

- *розрядність МП*. Іноді цей параметр називають внутрішньою розрядністю МП або розрядністю внутрішньої шини даних (ШД). Фактично ця кількість біт інформації, що одночасно можуть бути записані в кожній з регістрів загального призначення МП. Процесори 8086/8088 і 80286 були 16-розрядними, усі наступні МП цього сімейства, аж до Pentium IV, є 32-розрядними.

- *розрядність шини даних МП*. Іноді цей параметр називають зовнішньою розрядністю МП або розрядністю зовнішньої ШД. Це кількість зовнішніх виводів МП для передачі даних. Розрядність і внутрішньої і зовнішньої ШД можуть відрізнятися. Так, наприклад, МП першого покоління сімейства Intel x86 8086 і 8088 відрізнялися тим, що 8088 мав вісьмирозрядну зовнішню ШД, а 8086 - шістнадцятирозрядну, хоча розрядність внутрішньої ШД і в того, і в іншого дорівнювала 16.

- *розрядність шини адреси (ША)*. Це кількість зовнішніх виводів МП для передачі адресної інформації. Розрядність адреси прямо визначає максимальний обсяг фізично адресованої пам'яті як $M=2^N$, де N - розрядність ША, M - обсяг фізично адресованої пам'яті в байтах.

- *максимальний обсяг фізично адресованої пам'яті*. Цей параметр прямо зв'язаний з розрядністю ША МП.

- *швидкодія (продуктивність)*. Це інтегральна характеристика МП, до визначення якої існують різні підходи. По-перше, швидкодія МП істотна залежить від тактової частоти. Підвищення тактової частоти МП однозначно веде до підвищення продуктивності МП, хоча для обчислювальної системи в цілому ця залежність не є лінійною. Крім того, МП різних поколінь, що працюють на одній і тій же тактовій частоті мають істотно різну продуктивність (наприклад, Pentium з тактовою частотою 100 МГЦ, перевершує МП 486 з тією же тактовою частотою по продуктивності в 2-2,5 рази). Це визначається саме розходженнями в мікроархітектурі МП. По-друге, МП використовують дві істотно різних форми представлення чисел: з плаваючою і фіксованою комами (точками). Перші виконуються переважно блоком виконання операцій із точкою, що плаває, (математичнимспівпроцесором), а другі - АЛП самого МП. Тому прийнято роздільно оцінювати швидкість виконання МП операцій з числами, представленими в тій і іншій формах. Швидкість виконання цілочисленних операцій (над числами з фіксованою крапкою) виміряється в **MIPS**, швидкість виконання операцій із крапкою, що плаває - у **MFLOPS** відповідно. Крім того, варто враховувати, що продуктивність різних обчислювальних систем, побудованих на тому самому МП, може істотно розрізнятися через архітектурні розходження цих систем.

1.3. Принципи функціонування універсальних мікропроцесорів

На основі навчального матеріалу, що був розглянутий у попередній темі, можна сформулювати деякі загальні принципи побудови та функціонування універсальних МП:

1.Всі МП працюють у двійковій системі числення.

2.Будь-який МП складається з двох основних частин: пристрою управління та АЛП.

3.АЛП універсального МП будується на основі багаторозрядного суматора; операція додавання чисел із знаками – основна операція, що виконує АЛП. Додавання чисел із знаками звичайно виконується з використанням додаткового коду.

4.Обчислювальний процес організовується на основі принципу мікропрограмного управління.

Будь-який МП має у своєму складі набір регістрів різного призначення, частина яких доступна для дій, обумовлених безпосередньо програмістом - збереження операндів, виконання дій над ними і формування адрес і операндів у пам'яті. Ці регістри називаються регістрами загального призначення. Інша частина регістрів використовується процесором для службових (системних) цілей, доступ до них може бути обмежений. Є навіть програмно-невидимі регістри. Поняття *архітектури МП* визначає його складові частини, зв’язки та взаємодію між ними. Архітектурамістить: 1) структурну схему МП; 2) програмну модель МП (опис функцій регістрів); 3) організацію пам’яті (ємність пам’яті та способи її адресації); 4) опис організації процедур введення-виведення. Але, як правило, говорячи про *архітектуру МП*, звичайно мають на увазі його програмну модель, тобто програмно-видимі властивості. Зокрема, 32-розрядні процесори сімейства Intel x86 мають архітектуру IA-32 (Intel Architecture 32 bit).

Під *мікроархітектурою МП* розуміють внутрішню реалізацію цієї програмної моделі. Тобто, процесори, що мають ті самі програмно-видимі властивості (архітектуру), можуть істотно розрізнятися мікроархітектурними реалізаціями цих властивостей, оскільки розроблювачі МП безупинно прагнуть до максимального підвищення продуктивності, тобто швидкості виконання програм.

Спрощена схема мікропроцесора, зображена на рис. 2, містить у собі наступні складові частини:

- арифметико-логічний пристрій,
- пристрій керування,
- блок декодування команд,
- блок внутрішніх регістрів МП,
- пристрій шинного інтерфейсу, що включає в себе блок випереджальної вибірки,
- роздільні блоки буферної пам'яті (кеш-пам'яті) команд і даних.

SHAPE * MERGEFORMAT

АЛП

Пристрій керування

Блок внутрішніх регістрів

Блок декодування команд

Кеш даних

Кеш команд

Пристрій шинного інтерфейсу (Блок випереджальної вибірки)

Рис. 1.2. Спрощена схема універсального мікропроцесора

АЛП – це комбінаційна схема на основі суматора, який сигналами з виходів пристрою керування налагоджується на виконання певної арифметичної або логічної операції над операндами, які пересилаються з пам'яті або регістрів МП.

Пристрій керування відповідно до кодів команд та зовнішніх керуючих сигналів і сигналів синхронізації виробляє сигнали управління для всіх блоків МП.

Внутрішні регістри призначенні для зберігання проміжних результатів обчислень. У складі внутрішніх регістрів виділяється *акумулятор* – регістр у якому зберігається один з операндів. Після виконання команди в акумуляторі замість операнда розміщується результат операції.

Блок декодування команд формує сигнали для пристрою керування згідно з дешифрованим кодом команди.

Блок випереджальної вибірки самостійно ініціює випереджаючу вибірку кодів команд з пам'яті у чергу команд.

Кеш команд (даних) – буферна статична пам'ять, звернення до якої відбувається на частоті функціонування процесора.

Послідовність функціонування мікропроцесора визначається програмним кодом.

Програмний код - це послідовність команд або інструкцій, кожна з яких певним чином закодована і розташована в цілому числі суміжних байт пам'яті. Кожна інструкція обов'язково має *операційну* частину, що несе процесорові інформацію про необхідні дії. Операндна частина вказує процесорові, де знаходиться його "предмет праці" -- операнди. Операнди – це об'єкти у вигляді значення даних, вмісту регістрів або комірок пам'яті, з яким оперує команда.

Одноядерний процесор фактично може виконувати тільки один процес - передачу керування від інструкції до інструкції відповідно до програми яка виконується. При цьому можуть виконуватися переходи, розгалуження і виклики процедур, але весь цей ланцюжок запрограмований розроблювачем програми. Послідовність виконання інструкцій, запропонована програмним кодом, може бути порушена під впливом внутрішніх або зовнішніх (щодо процесора) причин - виключень і апаратних переривань.

Типова послідовність роботи МП при виконанні якої-небудь команди містить у собі наступні етапи:

- вибірку чергової команди і даних, необхідних для її виконання, з оперативної пам'яті ЕОМ,
- декодування команди,
- власне її виконання,
- запис отриманого результату в оперативну пам'ять.

Щоб зрозуміти, як працює МП, розглянемо крок за кроком, як він виконує нескладну задачу додавання двох чисел (2+3=5). Вирішується вона в чотири етапи.

Етап 1. Введення цифри "2" приводить МП у стан готовності і подає блокові попередньої вибірки сигнал на запит в оперативній пам'яті комп'ютера інструкції у відношенні нових даних , що надійшли,, оскільки командна кеш-пам'ять такої інструкції не містить.

Нова інструкція з роботи з даними надходить з пам'яті комп'ютера через шинний інтерфейс у МП і записується в командну кеш-пам'ять, де їй привласнюється код "2=X" Слідом за цим блок попередньої вибірки запитує з кеш-пам'яті копію коду "2=X", що направляє для подальшої обробки в блок декодування.

Блок декодування декодує інструкцію "2=X", перетворює її в ланцюжок двійкових символів, що пересилається в керуючий блок і в кеш даних, даючи їм указівка про те, як з отриманою інструкцією надходити далі.

Оскільки блоком декодування прийняте рішення про збереження цифри 2 у кеш даних, керуючий блок виконує відповідну інструкцію для коду "2=X": цифрі 2 у кеш-пам'яті даних привласнюється адреса "X", тут вона і буде знаходитися в чеканні подальших указівок.

Після цього копія коду "3=Y" надходить з командної кеш-пам'яті в блок попередньої вибірки, відкіля переправляється в декодувальний блок для подальшої обробки.

Етап 2. Введення цифри "3". Блок попередньої вибірки одержує команду на запит у системній пам'яті комп'ютера й у командній кеш-пам'яті інструкцій про дії у відношенні знову даних , що надійшли. Оскільки командна кеш-пам'ять таких інструкцій не містить, вони надійдуть з оперативної пам'яті.

Аналогічно команді "2=X," нові інструкції з даних надходять з пам'яті комп'ютера в МП і записуються в командну кеш-пам'ять, де одержують код адресації "3=Y".

Декодувальний блок декодує інструкцію "3=Y", перетворює її в ланцюжок двійкових символів, що направляє в керуючий блок і в кеш-пам'ять даних, даючи їм указівки про те, як з даною інструкцією поводитися далі.

Оскільки декодувальний блок приймає рішення про збереження цифри 3 у кеш-пам'яті даних, то він виконує відповідну інструкцію для коду "3=Y": цифрі 3 привласнюється в кеш-пам'яті даних адреса "Y", де вона, аналогічно цифрі 2, і буде знаходитися в чеканні подальших указівок.

Етап 3. Уведення команди "+" змушує блок попередньої вибірки запросити з оперативної пам'яті комп'ютера і командної кеш-пам'яті інструкції у відношенні знову даних, що надійшли. Як і в попередніх випадках, ці інструкції повинні бути отримані з оперативної пам'яті.

Оскільки мова йде про інструкцію, яка не використовувалася раніше, "плюс" надходить у МП із пам'яті комп'ютера і записується в командну кеш-пам'ять із присвоєнням адресного коду "X+Y=Z", що позначає операцію додавання.

Слідом за цим блок попередньої вибірки запитує з командної кеш-пам'яті копію коду "X+Y=Z" і пересилає її блокові декодування для подальшої обробки.

Блок декодування декодує інструкцію "X+Y=Z", перетворює її в ланцюжок двійкових символів, що направляє в керуючий блок і в кеш-пам'ять даних, даючи їм указівки про те, як з даною інструкцією надходити далі. Одночасно арифметичний логічний пристрій одержує вказівки на виконання операції ДОДАВАННЯ.

Блок керування аналізує код, а арифметичний логічний пристрій виконує по команді операцію ДОДАВАННЯ чисел, закодованих як "X" і "Y" і витягнутих з кеш-пам'яті даних. Після цього ALU, пересилає в блок регістрів отримане число 5 для запису по одному з адрес.

Етап 4. Вивід результату. Блок попередньої вибірки в черговий раз перевіряє командну кеш-пам'ять на предмет наявності інструкцій, що відносяться до нових даних , що надійшли. Як і колись, такі інструкції там відсутні. Інструкція для значка "=" надходить у МП з оперативної пам'яті комп'ютера через шинний інтерфейс і записується в командну кеш-пам'ять, одержавши код адресації "Print Z" ("відобразити на екрані символ Z).

Слідом за цим блок попередньої вибірки запитує з командної кеш-пам'яті копію коду "Print Z", що пересилає декодувальному блокові для подальшої обробки.

Декодувальний блок декодує інструкцію "Print Z", перетворює її в ланцюжок двійкових символів, що потім пересилається керуючому блокові з указівкою на те, як з отриманою інструкцією поводитися далі.

Тепер, коли значення величини, представленої кодом Z, вже обчислене і записано в позиції № 5 блоку регістрів, для завершення операції додавання 2+3 залишається виконати команду виводу вмісту регістра 5 на екран дисплея. На цьому робота МП закінчується.

ВИСНОВОК

Мікропроцесором у кібернетиці називають програмно - керований пристрій обробки інформації, виконаний на одній ВІС, чи на деякому наборі ВІС. Мікропроцесор (МП) є основним обчислювальним блоком комп'ютера. Це пристрій, що виконує програму - послідовність команд (інструкцій), написану програмістом і оформлену у вигляді модуля, що виконується.

По призначенню розрізняють універсальні і спеціалізовані МП.

По числу ВІС, використовуваних для побудови функціонально повного МП, розрізняють однокристальні, багатокристальні, багатокристальні секційні МП.

По розрядності оброблюваних кодових слів даних МП можуть бути з фіксованою або нарощуваною розрядністю слів.

По способу керування обчислювальним процесом розрізняють МП із мікропрограмним і з жорстким (апаратним) керуванням.

Найбільш практично важливими для кінцевого користувача конструктивними і функціональними характеристиками МП є: внутрішня розрядність, розрядність шини даних, розрядність шини адреси, максимальний обсяг фізично адресованої пам'яті; швидкодія (продуктивність).

ТЕМА 2. АРИФМЕТИЧНІ ОСНОВИ МІКРОПРОЦЕСОРНОЇ ТЕХНІКИ.

2.1. Системи числення.

Сукупність прийомів та правил найменування й позначення чисел називається **системою числення**. Звичайною для нас і загальноприйнятою є позиційна десяткова система числення. Як умовні знаки для запису чисел вживаються цифри.

Система числення, в якій значення кожної цифри в довільному місці послідовності цифр, яка означає запис числа, не змінюється, називається непозиційною. Система числення, в якій значення кожної цифри залежить від місця в послідовності цифр у записі числа, називається позиційною.

Щоб визначити число, недостатньо знати тип і алфавіт системи числення. Для цього необхідно ще додати правила, які дають змогу за значеннями цифр встановити значення числа.

Найпростішим способом запису натурального числа є зображення його за допомогою відповідної кількості паличок або рисочок. Таким способом можна користуватися для невеликих чисел.

Наступним кроком було винайдення спеціальних символів (цифр). У непозиційній системі кожен знак у запису незалежно від місця означає одне й те саме число. Добре відомим прикладом непозиційної системи числення є римська система, в якій роль цифр відіграють букви алфавіту: I - один, V - п'ять, X - десять, C - сто, Z - п'ятдесят, D -п'ятсот, M - тисяча. Наприклад, **324**= CCCXXIV. У непозиційній системі числення незручно й складно виконувати арифметичні операції.

2.2. Позиційна система числення.

Загальноприйнятою в сучасному світі є десяткова позиційна система числення, яка з Індії через арабські країни прийшла в Європу. Основою цієї системи є число десять. Основою системи числення називається число, яке означає, у скільки разів одиниця наступного розрядку більше за одиницю попереднього.

Загальновживана форма запису числа є насправді не що інше, як скорочена форма запису розкладу за степенями основи системи числення, наприклад

$$\mathbf{130678=1*10^5+3*10^4+0*10^3+6*10^2+7*10^1+8}$$

Тут 10 є основою системи числення, а показник степеня - це номер позиції цифри в записі числа (нумерація ведеться зліва на право, починаючи з нуля). Арифметичні операції у цій системі виконують за правилами, запропонованими ще в середньовіччі. Наприклад, додаючи два багатозначних числа, застосовуємо правило додавання стовпчиком. При цьому все зводиться до додавання однозначних чисел, для яких необхідним є знання таблиці додавання.

Проблема вибору системи числення для подання чисел у пам'яті комп'ютера має велике практичне значення. В разі її вибору звичайно враховуються такі вимоги, як надійність подання чисел при використанні

фізичних елементів, економічність (використання таких систем числення, в яких кількість елементів для подання чисел із деякого діапазону була б мінімальною). Для зображення цілих чисел від **1** до **999** у десятковій системі достатньо трьох розрядів, тобто трьох елементів. Оскільки кожен елемент може перебувати в десятьох станах, то загальна кількість станів - **30**, у двійковій системі числення $\mathbf{999_{10}=1111100}$, необхідна кількість станів - **20** (індекс знизу зображення числа - основа системи числення). У такому розумінні є ще більшекономічна позиційна система числення - трійкова. Так, для запису цілих чисел від 1 до у десятковій системі числення потрібно**90** станів, у двійковій - **60**, у трійковій - **57**. Але трійкова система числення не дістала поширення внаслідок труднощів фізичної реалізації.

Тому найпоширенішою для подання чисел у пам'яті комп'ютера є двійкова система числення. Для зображення чисел у цій системі необхідно дві цифри: 0 і 1, тобто достатньо двох стійких станів фізичних елементів. Ця система є близькою до оптимальної за економічністю, і крім того, таблички додавання й множення в цій системі елементарні.

Оскільки $\mathbf{2^3=8}$**, а** $\mathbf{2^4=16}$, то кожних три двійкових розряди зображення числа утворюють один вісімковий, а кожних чотири двійкових розряди - один шістнадцятковий. Тому для скорочення запису адрес та вмісту оперативної пам'яті комп'ютера використовують шістнадцяткову й вісімкову системи числення.

В процесі налагодження програм та в деяких інших ситуаціях у програмуванні актуальною є проблема переведення чисел з однієї позиційної системи числення в іншу. Якщо основа нової системи числення дорівнює деякому степеню старої системи числення, то алгоритм переводу дуже простий: потрібно згрупувати справа наліво розряди в кількості, що дорівнює показнику степеня і замінити цю групу розрядів відповідним символом нової системи числення. Цим алгоритмом зручно користуватися коли потрібно перевести число з двійкової системи числення у вісімкову або шістнадцяткову. Наприклад, $\mathbf{10110_2=10\ 110=26_8}$,$\mathbf{1011100_2=101\ 1100=5C_8}$

у двійковому відбувається за зворотнім правилом: один символ старої системи числення заміняється групою розрядів нової системи числення, в кількості рівній показнику степеня нової системи числення. Наприклад, $\mathbf{472_8=100\ 111\ 010=100111010_2}$**, B516=1011 0101=**$\mathbf{10110101_2}$

Як бачимо, якщо основа однієї системи числення дорівнює деякому степеню іншої, то перевід тривіальний. У протилежному випадкові користуються правилами переведення числа з однієї позиційної системи числення в іншу (найчастіше для переведення із двійкової, вісімкової та шістнадцяткової систем числення у десяткову, і навпаки).

2.3. Алгоритми переведення чисел з однієї позиційної системи числення в іншу.

Для переведення чисел із системи числення з основою **p** в систему числення з основою **q**, використовуючи арифметику нової системи числення з основою **q**, потрібно записати коефіцієнти розкладу, основи степенів і

показники степенів у системі з основою **q** і виконати всі дії в цій самій системі. Очевидно, що це правило зручне при переведенні до десяткової системи числення.

Наприклад:

з шістнадцяткової в десяткову:

$\mathbf{92C8_{16}=9*10_{16}^{3}+2*10_{16}^{2}+C*10_{16}^{1}+8*10_{16}^{0}=}$

$\mathbf{9*16_{10}^{3}+2*16_{10}^{2}+12*16_{10}^{1}+8*16_{10}^{0}=37576}$

з вісімкової в десяткову:

$\mathbf{735_{8}=7*10_{8}^{2}+3*10_{8}^{1}+5*10_{8}^{0}= 7*8_{10}^{2}+3*8_{10}^{1}+5*8_{10}^{0}=477_{10}}$

з двійкової в десяткову:

$\mathbf{110100101_{2}=1*10_{2}^{8}+1*10_{2}^{7}+0*10_{2}^{6}+1*10_{2}^{5}+0*10_{2}^{4}+0*10_{2}^{3}+1*10_{2}^{2}+0*10_{2}^{1}+1*10_{2}^{0}=1*2_{10}^{8}+1*2_{10}^{7}+0*2_{10}^{6}+1*2_{10}^{5}+0*2_{10}^{4}+0*2_{10}^{3}+1*2_{10}^{2}+0*2_{10}^{1}+1*2_{10}^{0}=421_{10}}$

Для переведення чисел із системи числення з основою **p** в систему числення з основою **q** з використанням арифметики старої системи числення з основою **p** потрібно:

- для переведення цілої частини:
  - послідовно число, записане в системі основою **p** ділити на основу нової системи числення, виділяючи остачі. Останні записані у зворотному порядку, будуть утворювати число в новій системі числення;
- для переведення дробової частини:
  - послідовно дробову частину <u>множити</u> на основу нової системи числення, виділяючи цілі частини, які й будуть утворювати запис дробової частини числа в новій системі числення.

Цим самим правилом зручно користуватися в разі переведення з десяткової системи числення, тому що її арифметика для нас звичніша.

2.4. Форми подання двійкових чисел.

Двійкові числа в обчислювальних пристроях розміщуються у комірках пам'яті, причому для кожного розряду числа виділяється окрема комірка, що зберігає один біт інформації. Сукупність комірок, призначених для розміщення одного двійкового числа, називають *розрядною сіткою*. Довжина розрядної сітки (число комірок *n* у розрядній сітці) обмежена і залежить від конструктивних особливостей обчислювального пристрою. Більшість існуючих електронних обчислювальних пристроїв мають розрядні сітки, що містять 16, 32 або 64 комірок.

Розміщення розрядів числа у розрядній сітці може відбуватися різними способами. Спосіб розміщення визначається формою подання двійкових чисел у ЕОМ. ***Розрізняють дві форми подання двійкових чисел: із фіксованою комою і з «плавучою» комою.*** Іноді ці форми називаютьвідповідно *природною і напівлогарифмічною*.

Припустимо, що в розрядній сітці необхідно розмістити двійкове число, що містить цілу і дробову частини. Якщо для розміщення цілої

частини числа виділяється k комірок n-розрядної сітки, то (якщо не враховувати знак) для розміщення дробової частини залишиться n-k вільних комірок (рис. 1).

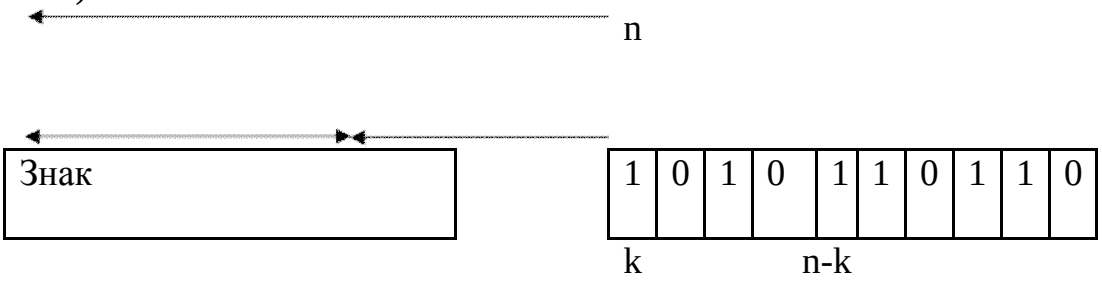


Рис. 1. Форма подання двійкових чисел із фіксованою комою.

Така форма подання двійкових чисел називається *формою з фіксованою комою.* Дійсно, положення коми строго фіксовано стосовно розрядної сітки. Якщо кількість розрядів у дробовій частині числа перевищують n-k, то деякі молодші розряди виходять за межі розрядної сітки і не будуть сприйматися обчислювальним пристроєм. ***Отже, будь-яке двійкове число, менше ніж одиниця молодшого розряду розрядної сітки, сприймається як нуль і називається машинним нулем.***

У результаті відкидання молодших розрядів дробової частини числа, розташованої за межами розрядної сітки, виникає похибка подання. Максимальне значення абсолютної похибки подання не перевищує одиниці молодшого розряду сітки.

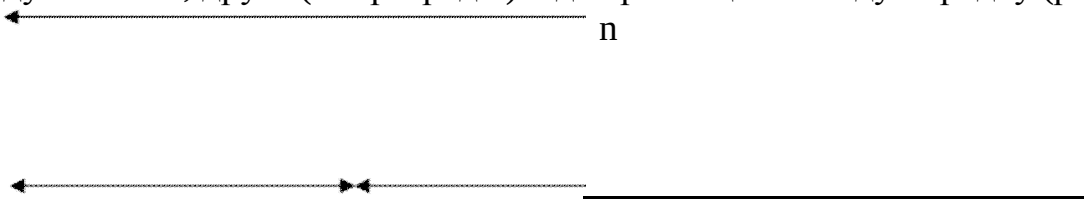
В універсальних ЕОМ форма з фіксованою комою, у зв'язку з властивою їй низькою точністю, застосовується лише для подання цілих чисел. Основною є форма подання чисел з «плавучою» комою. Її використання дозволяє суттєво розширити діапазон і зменшити відносну похибку.

У цій формі числа подаються у вигляді суми деякого ступеня основи системи числення (який називається характеристикою числа) і цифрової частини, що має вигляд правильного дробу:

$$N = \pm a q^{\pm p},$$

де p звуть порядком числа, а правильний дріб a – його мантисою. Мантиса і порядок є знаковими числами. Тому для позначення знаків у розрядній сітці відводяться два додаткові розряди. Знак усього числа співпадає із знаком мантиси.

При запису двійкового числа у показовій формі, в розрядній сітці використовуються дві групи розрядів (без урахування знакових розрядів мантиси і порядку). Перша група (k розрядів) призначена для розміщення коду мантиси, друга (n-k розрядів) – для розміщення коду порядку (рис.2).



Знак мантиси	1	0	1	0	1	1	0	1	1	0	Знакпорядку

k n-k

Рис. 2. Форма подання двійкових чисел із „плавучою” комою.

Отже, мантиса числа може мати необмежену кількість різних значень, менших за одиницю, при відповідних значеннях порядку (тобто кома може «плавати»). ***З усієї кількості подань числа у показовій формі те його подання, що не має в старшому розряді мантиси нуля, називають нормалізованим.*** Всі інші подання є ненормалізованими. У нормалізованій формі значення мантиси завжди більші або дорівнюють 1/2, але не перевищують одиниці.

У обчислювальних пристроях із «плавучою» комою усі числа зберігаються у нормалізованому вигляді, при цьому не втрачаються молодші розряди мантиси і підвищується точність обчислень. Якщо після виконання будь-якої арифметичної операції результат виявляється ненормалізованим, то перед занесенням числа в пам’ять виконують його нормалізацію, тобто зсув мантиси ліворуч на відповідну кількість розрядів, і зменшення порядку числа на відповідну кількість одиниць.

Показова форма подання чисел має і свої вади, основною з яких є порівняно висока складність виконання арифметичних операцій, а отже, і більша вимогливість до ресурсів обчислювального пристрою. Це обмежує її застосування, наприклад, у спеціалізованих радіотехнічних обчислювальних пристроях, у системах управління технологічними процесами та обробки вимірювальної інформації у реальному часі.

2.5. Прямий, обернений і додатковий коди двійкових чисел.

Залежно від способу обробки бітів, розміщених у розрядній сітці, розрізняють два види кодів: паралельний, коли в кожний момент часу всі розряди сітки доступні для обробки, і послідовний, коли в кожний момент часу доступний один розряд сітки. Числа, подані паралельним кодом, доступні за один такт, а числа, подані послідовним кодом, – за n тактів, де n – розрядність сітки. Якщо розрядність числа перевищує довжину сітки, то йогообробка ведеться частинами.

Натуральним кодом називають подання числа як цілого беззнакового у двійковій системі числення. Діапазон подання чисел у натуральному коді для n- розрядної сітки становить від 0 до 2^n-1, тобто для 8-розрядної сітки – від 0 до 255.

Для подання цілих знакових чисел використовують прямий, обернений і додатковий коди. Старший розряд сітки є знаковим. Значення цього розряду дорівнює 0 для додатних чисел і 1 – для від’ємних. В інших розрядах розміщується модуль числа.

Якщо до натурального коду цілого числа додати знаковий розряд, одержуємо запис числа у прямому коді (ПК). Домовимося знаковий розряд розташовувати зліва і відокремлювати від розрядів модуля числа крапкою, наприклад: $+ 6_{(10)} = 0.\ 110_{(ПК)}$; $- 6_{(10)} = 1.\ 110_{(ПК)}$.

Використання ПК забезпечує виконання операції додавання двох додатніх чисел звичайним способом без будь-яких складностей – не варто лише робити перенос одиниці старшого розряду модуля суми у знаковий розряд. Тобто при виконанні арифметичних операцій над ПК двійкових чисел знаковий розряд і розряди модуля не можна розглядати як єдине ціле. У цьому можна переконатися, розглянувши такий приклад:

Правильно:	Неправильно:
0. 0110	0.0110
+ 0. 1010	+ 0.1010
0.10000	1.0000
(+6) + (10) = (+16)	(+6) + (+10) = (- 0)

Однак, виконання операції віднімання одного числа від іншого шляхом безпосереднього додавання їхніх ПК неможливо. Неважко також помітити, що в ПК нуль має два можливі зображення: - 0 = 1.000... і + 0 = 0.000..., що ускладнює інтерпретацію результатів виконання арифметичних операцій у ЕОМ.

Іншою формою запису двійкових чисел є обернений код (ОК).

ОК двійкового від'ємного числа утворюється з ПК рівного йому за модулем додатнього числа шляхом інвертування значень усіх його розрядів. Або: ***ОК від'ємного числа утворюється шляхом інверсії всіх розрядів модуля цього числа, записаного у ПК. Знаковий розряд при цьому зберігає значення 1.*** Наприклад, $6_{(10)} = 1.110_{(ПК)} = 1.001_{(ОК)}$.

При виконанні арифметичних операцій над двійковими числами, поданими в ОК, знаковий розряд і розряд модуля числа можна розглядати як єдине ціле (перенос одиниці зі старшого розряду модуля суми в знаковий розряд не приводить до помилкового результату), але нуль як і раніше має два зображення – «додатнє» і «від'ємне». Слід зазначити, що отриманий при додаванні від'ємний результат також утворюється в ОК. У цьому випадку число може бути перетворене у ПК інверсією всіх значущих розрядів (розрядів модуля). Наприклад:

$$\begin{array}{r} 0.110 \\ +\underline{1.001} \\ 1.111_{(ОК)} = 1.000_{(ПК)} \end{array}$$

(+6) + (-6) = (-0)

Найбільше поширення в обчислювальних пристроях одержало подання від'ємних двійкових чисел за допомогою додаткового коду (ДК).

ДК від'ємного числа утворюється з його прямого коду за правилом:

- ***у знаковому розряді залишається одиниця;***
- ***розряди модуля числа інвертуються;***
- ***до молодшого розряду додається одиниця.***

Очевидно, що ДК від'ємного числа утворюється з його ОК додаванням одиниці до молодшого розряду.

Наприклад, $- 6_{(10)} = 1.010_{(ДК)}$.

Дійсно, для числа - 6 маємо:

$$\begin{array}{r} 1.110_{(ПК)} \\ 1.001_{(ОК)} \\ +\quad 1 \\ \hline 1.010_{(ДК)}. \end{array}$$

Зворотний перехід від ДК до ПК або ОК відбувається за тими ж правилами.

Головною перевагою ДК є те, що цифра 0 у ньому має єдине подання: 0.000... Саме тому, для подання від'ємних чисел у сучасних ПЕОМвикористовується переважно ДК.

Неправильний дріб (число, що має цілу частину) із знаком записують у різних кодах за допомогою традиційного роздільника – коми між цілою і дробовою частиною. Наприклад: - $118{,}375_{(10)}$ = $1.0001{,}101_{(ДК)}$.

Слід пам'ятати, що для кодування додатніх чисел застосовується тільки ПК, хоча можна сказати, що для таких чисел ДК і ОК збігаються з прямим.

Операція одержання ДК від'ємного числа з ПК рівного йому за модулем додатнього числа називається операцією доповнення. Ця операціяполягає в інвертуванні всіх розрядів вихідного коду (включаючи знаковий) і додавання до молодшого розряду одиниці.

Таким чином, сформулюємо наступне правило***: у системі двійкових чисел із знаком заміна додатнього числа на рівне йому за модулем від'ємне і навпаки, від'ємного на додатнє, здійснюється шляхом застосуванням до коду даного числа операції доповнення.***

Ця властивість подання від'ємних чисел у ДК дозволяє при виконанні арифметичних операцій взагалі відмовитися від операції віднімання, замінивши її операцією додавання з числом, що має знак, протилежний знаку числа, яке віднімається.

2.6. Алгоритми виконання арифметичних операцій над двійковими числами із знаком.

Додавання двійкових чисел із знаком

Очевидно, що при додаванні чисел із знаком можуть виникати переноси одиниці із старшого розряду модуля суми до знакового розряду (домовимося позначати його P_1) та із знакового розряду – ліворуч за межі розрядної сітки, у розряд переповнення (P_2). Через використання розглянутих раніше кодів, у яких знак числа позначається тими ж цифрами, що і розряди модуля, переповнення розрядної сітки може виникати навіть у випадку додавання чисел із різними знаками, коли модуль результату не перевищує модуля будь-якого операнда. При додаванні ж двох від'ємних чисел перенесення одиниці до розряду переповнення відбувається завжди.

При виникненні переповнення розрядної сітки для одержання правильного результату додавання необхідно застосовувати таке правило:

· ***якщо P_1 Å P_2 = 0, одиниця в розряді переповнення ігнорується (відкидається);***

· ***якщо P1 Å P2 = 1, необхідно зсунути число на один розряд праворуч (або зсунути позицію точки на один розряд ліворуч).***

Додавання дробових і цілих двійкових чисел, поданих у формі з фіксованою комою, відбувається однаково, тобто порядок додавання не залежить від розташування коми. Тому операцію додавання розглянемо на прикладі додавання цілих чисел.

Приклади:

1) Додавання двох додатніх чисел (без переповнення розрядної сітки).

$$\begin{array}{r} 0.100111 \\ +\ 0.001101 \\ \hline 0.110100 \end{array} \qquad \begin{array}{r} 39 \\ +13 \\ \hline 52 \end{array}$$

P_1Å P_2 = 0 – результат коректний і остаточний.

2) Додавання двох додатніх чисел (з переповненням розрядної сітки).

$$\begin{array}{r} 0.01101 \\ + \\ \hline 0.10011 \\ \\ 1.00000 \end{array} \qquad \begin{array}{r} 13 \\ +19 \\ \hline 32 \end{array}$$

P_1 Å P_2 = 1. Результат некоректний, тому що відбулося переповнення розрядної сітки. Зсуваючи число на один розряд праворуч, остаточно маємо $0.100000_{(ПК)} = 32_{(10)}$.

3) Додавання двох чисел із різними знаками (без переповнення розрядної сітки)

$$\begin{array}{r} 1.001100 \\ +\ 0.001101 \\ \hline 1.011001 \end{array} \qquad \begin{array}{r} -\ 52 \\ +\ 13 \\ \hline -\ 39 \end{array}$$

P_1 Å P_2 = 0. Результат коректний, але тому що він є від’ємним, для перевірки правильності розв’язання необхідно перетворити його у прямий код. Остаточно маємо $1.100111_{(ПК)} = 39_{(10)}$.

4) Додавання двох чисел, рівних за модулем і різних за знаком.

$$\begin{array}{r} 1.011001 \\ +\ 0.100111 \\ \hline 10.000000 \end{array} \qquad \begin{array}{r} -\ 39 \\ +\ 39 \\ \hline 0 \end{array}$$

P_1Å P_2 = 0. Результат коректний, якщо не брати до уваги одиницю у розряді переповнення.

Додавання двох від’ємних чисел виконується аналогічно прикладам 1, 2 (у залежності від значення виразу P_1 Å P_2. Тому що результат у цьому випадку завжди від’ємний, для перевірки правильності розв’язання необхідно перетворити його у прямий код, аналогічно прикладу 3.

Висновки:

· Правильність виконання операцій додавання обов'язково повинна перевірятися шляхом аналізу значення виразу P_1Å P_2, щоб уникнути одержання некоректного результату, що виникає при переповненні розрядної сітки, при цьому: якщо P_1 Å P_2 = 0, одиниця в розряді переповнення

ігнорується (відкидається); якщо P_1Å P_2 = 1, необхідно зсунути число на один розряд праворуч.

· Правило перевірки коректності результату додавання двійкових чисел також можна сформулювати в такий спосіб: якщо знак операндів однаковий, а знак суми протилежний, результат є некоректним. При додаванні двох операндів із різними знаками результат завжди коректний, якщо не брати до уваги одиницю у розряді переповнення.

Множення і ділення двійкових чисел із фіксованою комою

Множення двійкових чисел завжди виконують у прямому коді. Знак добутку визначають по знакових розрядах множників згідно з таким загальновідомим правилом: ***якщо знаки операндів однакові, то знак добутку – позитивний; у протилежному випадку – знак добутку негативний.***

Знак добутку двох чисел не впливає на алгоритм виконання операції множення модулів цих чисел.

Часто використовують спосіб множення, процедура якого аналогічна процедурам множення вручну. У цьому випадку результат одержують додаванням часткових добутків. Кожний частковий добуток удвічі перевищує попередній, що відповідає його зсуванню ліворуч на один розряд. Наприклад:

```
    1101          13
  x 1011        x 11
    1101          13
   1101        + 13
+ 0000         ------ 143
1101
10001111
```

Характерно, що розрядність добутку двійкових чисел удвічі перевищує розрядність співмножників. Якщо у множенні беруть участь мантиси, тобто правильні дроби, то молодші розряди, що виходять за межі розрядної сітки, можуть бути відкинуті без округлення або з округленням.

Операція ділення також виконується способом, аналогічним застосовуваному при діленні вручну, що наочно ілюструє приклад ділення двох чисел 506 : 23 = 22, тобто 0.111111010 : 0.10111 = 0.10110. Знак частки визначають аналогічно знаку добутку. Застосоване при діленні віднімання дільника виконують шляхом додавання його додаткового коду.

```
          0. 1 1 1 1 1 1 0 1 0   ділене додатнє
          1. 0 1 0 0 1           перше віднімання
                                 дільника
1 1 0 0. 1 0 0 0 1               1 – результат додатній
          1. 0 1 0 0 1           – друге віднімання
                                 дільника
0         1 1. 1 0 1 0 0         0 – від'ємний результат
             0. 1 0 1 1 1        – додавання дільника
1         1 0 0. 1 0 1 1 1       1 – результат додатній
               1. 0 1 0 0 1      – третє віднімання
```

дільника

1 1 **0** 0. 0 0 0 0 1 – остача дорівнює нулю

0

У даному прикладі використаний так названий алгоритм без відновлення остачі, що передбачає таку послідовність дій:

· із діленого віднімається дільник (додається дільник, записаний у додатковому коді);

· якщо остача додатня, перша цифра частки дорівнює одиниці, у протилежному випадку – 0;

· остача зсувається ліворуч, і до неї додається дільник із знаком, зворотним знаку остачі;

· знак наступної остачі визначає наступну цифру частки;

· ці дії повторюють доти, поки не утвориться необхідне число розрядів частки або нульова остача.

Слід зазначити, що оскільки даний алгоритм передбачає додавання чисел (остач і дільника) тільки з протилежними знаками, то всі розряди проміжних сум, старші за знаковий, слід ігнорувати.

Виконання арифметичних операцій у пристроях із «плавучою» комою

Операція додавання у пристроях із «плавучою» комою відбувається у чотири етапи:

1.Порівнюються порядки доданків: менший порядок збільшується до більшого. При цьому відповідним чином корегується мантиса числа, яке перетворюється.

2.Виконується перетворення мантис у додаткові коди.

3.Виконується додавання мантис за правилами, розглянутими вище для чисел із фіксованої комою.

4.До суми приписується порядок доданків і, в разі необхідності, виконується нормалізація результату.

Операція множення чисел, поданих у формі з «плавучою» комою також виконується у чотири етапи:

1.Визначається знак добутку.

2.Перемножуються мантиси співмножників за правилами для чисел із фіксованої комою.

3.Обчислюється порядок добутку алгебраїчним додаванням порядків співмножників за правилами додавання цілих чисел із знаком.

4.Виконується нормалізація отриманого результату у випадку її необхідності.

Ділення чисел у пристроях із «плавучою» комою виконується так само, як і множення.

ТЕМА 3. ЛОГІЧНІ ОСНОВИ.

3.1 Основні закони алгебри логіки та їх використання для подання одних функцій логіки через інші.

Вперше логічні функції були використані в алгебрі логіки, початок якій покладено працями англійського математика Дж. Буля, її також називають булевою алгеброю або алгеброю висловлень.

Під висловленням розуміється будь-яке твердження, яке може бути істинним або хибним.

Істинному висловленню приписується 1, хибному – 0. Висловлення можуть бути простими і складними. Складні висловлення складаються з простих.

Для об'єднання простих висловлень в складні використовуються логічні зв'язки, що відповідають логічним функціям, аргументами яких є прості висловлення.

Логічний зв'язок "І" (кон'юнкція). Кон'юнкцією називають складне висловлення, що містить 2 або більше простих висловлень і яке є істинним тоді і лише тоді, коли істинними є прості висловлення, і хибним, якщо хоч одне з простих висловлень хибне.

Кон'юнкція являє собою логічний зв'язок "І" (див. табл. 3.1).

З'єднання двох висловлень читається як "x і y". Позначається xy або $x \wedge y$.

Таблиця 3.1

x	0	0	1	1
y	0	1	0	1
$x \cdot y = x \wedge y$	0	0	0	1

Логічний зв'язок "АБО" (диз'юнкція). Диз'юнкцією називають складне висловлення, що містить декілька простих висловлень і яке є істинним тоді, коли істинним буде хоч одне з простих висловлень, які входять в це складне висловлення, і хибним, якщо всі прості висловлення хибні.

Диз'юнкція являє собою логічний зв'язок "АБО" (табл. 3.2) і позначається $x \vee y$. Читається "x або y".

Таблиця 3.2

x	y	$x \vee y$ = "x або y"
0	0	0
0	1	1
1	0	1
1	1	1

Логічний зв'язок "НЕ" (заперечення). Логічний зв'язок "НЕ" означає заперечення висловлення і читається "НЕ x", позначається $\overline{x}$ абоШx (табл. 3.3)

Таблиця 3.3

x	0	1
$\overline{x}$	1	0

Запереченням висловлення x називають складне висловлення “НЕ x”, яке є істинним, коли x хибне, і хибним, коли x істинне.

Для зручності подальших викладок використаємо позначення: “$\cdot$” – кон’юнкція, “$\vee$” – диз’юнкція і “ $\overline{\ }$ ” – заперечення.

Булевою алгеброю називається множина M, що складається не менше ніж з двох елементів, на якій визначені три операції – диз’юнкції ($x \vee y$), кон’юнкції (xy), заперечення ($\overline{x}$). Для будь-яких елементів $x, y, z \in M$ виділяємо набір незалежних властивостей, які вважають аксіомами булевої алгебри, а саме:

– закон комутативності:

$$\left.\begin{array}{l} x \vee y = y \vee x \\ x \cdot y = y \cdot x \end{array}\right\}; \qquad (1.1)$$

– закон асоціативності:

$$\left.\begin{array}{l} (x \vee y) \vee z = x \vee (y \vee z) \\ (xy)z = x(yz) \end{array}\right\};$$

– закон дистрибутивності:

$$\left.\begin{array}{l} x(y \vee z) = xy \vee xz \\ x \vee (y \cdot z) = (x \vee y) \cdot (x \vee z) \end{array}\right\};$$

для спрощення формул крім аксіом використовують такі співвідношення або закони алгебри логіки:

– логічне додавання до нуля:

$$x \vee 0 = x$$

– логічне додавання до одиниці:

$$x \vee 1 = 1;$$

– логічне множення на 0:

$$x \cdot 0 = 0;$$

– логічне множення на 1:

$$x \cdot 1 = x;$$

– закон протиріччя:

$$x \cdot \overline{x} = 0;$$

– закон виключеного третього:

$$x \vee \overline{x} = 1.$$

Всі інші закони є наслідком зазначених вище:

– закон ідемпотентності:

$$\left.\begin{array}{l} x \vee x \vee x = x \\ x \cdot x \cdot x = x \end{array}\right\};$$

– закон подвійного заперечення:

$$\overline{\overline{x}} = x;$$

– закон поглинання (*x* поглинає *y*):

$$x \vee xy = x$$
$$(x \vee y)x = x;$$

– закон де Моргана:

$$\overline{x \vee y} = \overline{x}\,\overline{y};$$
$$\overline{xy} = \overline{x} \vee \overline{y};$$

– наслідки законів де Моргана:

$$x \vee y = \overline{\overline{x} \wedge \overline{y}};$$
$$xy = \overline{\overline{x} \vee \overline{y}}.$$

За допомогою розглянутих співвідношень можна виконувати різні тотожні перетворення булевих виразів.

При цьому порядок виконання дій такий:

При відсутності дужок виконуються операції заперечення, потім кон’юнкції, останніми – диз’юнкції.

Подання одних функцій алгебри логіки через інші

1. Операція заборони:

$$x_1 \Delta x_2 = x_1 \cdot \overline{x_2}.$$

Для доведення цього і наступних співвідношень будемо підставляти в ліву і праву частини виразу окремі значення аргументів і перевіряти правильність рівності.

Таблиця 3.4

		x_1
		0
		0
		1
		0

Таблиця 3.5

			x_1
			0
			0
			1
			0

2. Сума за модулем 2:

$$x_1 \oplus x_2 = x_1 \cdot \overline{x_2} \vee \overline{x_1} \cdot x_2 = (x_1 \vee x_2) \cdot (\overline{x_1} \vee \overline{x_2}).$$

Таблиця 1.10

x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Таблиця 1.11

x_1	x_2	$x_1 \cdot \overline{x_2}$	$\overline{x_1} \cdot x_2$	$x_1 \cdot \overline{x_2} \vee \overline{x_1} \cdot x_2$
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

3. Операція Пірса:

$$x_1 \downarrow x_2 = \overline{x_1 \vee x_2}$$ (операція АБО-НЕ).

Таблиця 1.12

x_1	x_2	$x_1 \vee x_2$	$\overline{x_1} \vee \overline{x_2}$	$(x_1 \vee x_2) \cdot (\overline{x_1} \vee \overline{x_2})$
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

Таблиця 1.13

x_1	x_2	$x_1 \downarrow x_2$	$\overline{x_1 \vee x_2}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

4. Логічна рівнозначність:

$$x_1 \sim x_2 = \overline{x_1 \oplus x_2} = x_1 \cdot x_2 \vee \overline{x_1} \cdot \overline{x_2} = (\overline{x_1} \vee x_2) \cdot (x_1 \vee \overline{x_2}).$$

Справедливість першої рівності може бути встановлена безпосередньо по таблицях істинності функції логічної рівнозначності і суми по модулю 2; наступних рівностей - шляхом інвертування лівої і правої частин виразу і перетворення за формулами де Моргана.

5. Імплікація:

$$x_1 \to x_2 = \overline{x_1} \vee x_2.$$

Таблиця 1.14

x_1	x_2	$x_1 \to x_2$
0	0	1
0	1	1
1	0	0
1	1	1

Таблиця 1.15

x_1	x_2	$\overline{x_1}$	$\overline{x_1} \vee x_2$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

6. Функція Шеффера:

$$x_1 / x_2 = \overline{x_1 \cdot x_2}$$ (операція І-НЕ).

Таблиця 1.16

x_1	x_2	x_1/x_2
0	0	1
0	1	1
1	0	1
1	1	0

Таблиця1.17

x_1	x_2	$x_1 \cdot x_2$	$\overline{x_1 \cdot x_2}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

3.2. Мінімізація булевих функцій за допомогою карт Карно

Найважливішим допоміжним засобом для визначення найбільш простий логічної функції є карта Карно. Це не що інше, як змінена запис таблиці істинності. У цьому випадку значення аргументів не просто записуються поруч один з одним, а розміщуються по горизонталі і вертикалі таблиці, ділячи її, на зразок шахової дошки, на окремі квадрати. При парній кількості аргументів половину з них записують по горизонталі, а половину - по вертикалі. При непарному числі аргументів по горизонталі розміщується на один аргумент більше, ніж по вертикалі.

Порядок розміщення різних комбінацій значень аргументів слід вибирати таким, щоб при переході від одного осередку до сусідньої змінювався лише один аргумент. У ці осередки заносять ті значення логічної функції, які відповідають значенням аргументів. Як приклад (рис. 3.1) наведена таблиця істинності і відповідна їй карта Карно для функції И (кон'юнкції).

A	*B*	*F*
0	0	0
0	1	0
1	0	0
1	1	1

а

B \ *A*	0	1
0	0	0
1	0	1

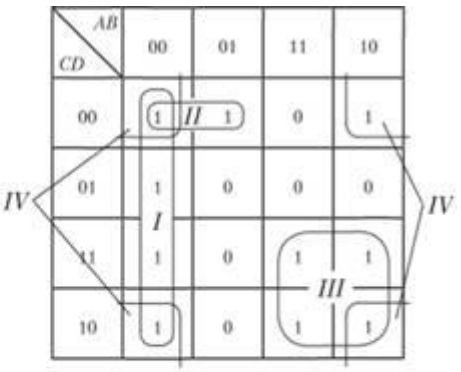
б

Рис. 3.1. Таблиця істинності *(а)* і відповідна їй карта Карно *(б)* для функції И (кон'юнкції)

Карта Карно є спрощеною формою запису таблиці істинності, тому на її основі можна скласти СовДНФ шуканої логічної функції, користуючись описаним вище методом. Перевагою карт Карно є простота виявлення можливих спрощень логічної функції. Розглянемо це на прикладі, представленому на рис. 3.2.

A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

а



б

Рис. 3.2. Таблиця істинності *(а)* і відповідна їй карта Карно *(б)* для функції *F*

У першу чергу при складанні СовДНФ слід, як зазначалося вище, скласти кон'юнкції всіх аргументів для кожної констітуенти одиниці, тобто для кожного осередку, в якій стоїть одиниця. Для клітинки, розташованої в лівому верхньому кутку, отримуємо:

$K' = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D};$

для клітинки, розташованої правіше:

$K'' = \overline{A}B\overline{C}\,\overline{D}.$

Коли буде складена диз'юнкція всіх констітуєнт одиниці, крім інших в ній зустрінеться і такий фрагмент:

$K' \vee K'' = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} \quad \overline{A}B\overline{C}\,\overline{D}.$

Він спрощується таким чином:

$K' \vee K'' = \overline{A}\,\overline{C}\,\overline{D}(\overline{B} \quad B) \vee \overline{A}\,\overline{C}\,\overline{D}.$

Звідси випливає загальне правило спрощення логічних функцій для карт Карно: якщо в двох, чотирьох, восьми і т.д. осередках, обмежених прямокутним або квадратним контуром, стоять тільки одиниці, можна записувати безпосередньо кон'юнкцію для всієї цієї групи, причому в неї повинні входити лише ті аргументи, які залишаються незмінними для всіх осередків даної групи.

Таким чином, у цьому прикладі кон'юнкція для групи ***II,*** що складається з двох осередків, дорівнює

$K_{II} = \overline{A}\,\overline{C}\,\overline{D},$

що відповідає раніше отриманої функції. В одну групу зв'язуються також ті осередки, які знаходяться на лівому і правому краях одного рядка або у верхній і нижній частинах одного стовпчика.

Для групи ***I,*** що складається з чотирьох клітинок, можна записати:

$K_I = \overline{A}\,\overline{B}.$

Для групи ***III,*** що має квадратну форму і складається також з чотирьох клітинок, отримаємо таку кон'юнкцію:

$K_{III} = AC.$

Ще одна одиниця залишилася в правому верхньому куті. Вона може бути пов'язана, наприклад, з одиницею в нижній частині розглянутого стовпця в групу з двох осередків. Інша можливість полягає в об'єднанні одиниць, що знаходяться на лівому і правому краях першого рядка. Однак якщо взяти до уваги, що в кожному кутку карти Карно знаходиться одиниця, то можна знайти просте рішення. Пов'язуючи ці одиниці в одну чотирьохелементний групу, отримаємо:

$$K_{IV} = \overline{B}\,\overline{D}.$$

Таким чином, за допомогою карти Карно можна знайти найпростіший варіант логічної функції ***F:***

$$F = K_I \vee K_{II} \vee K_{III} \vee K_{IV} = \overline{A}\,\overline{B} \vee \overline{A}\,\overline{C}\,\overline{D} \vee AC \vee \overline{B}\,\overline{D}.$$

3.3.Алгебра логіки та цифрові електронні схеми.

Виникає питання: як можна уявити логічні функції за допомогою електричних схем? Так як логічні змінні можуть мати тільки два дискретних значення, слід звернути увагу на схеми, які можуть знаходитися в двох легко помітних станах. Такими схемами є електричні перемикаючі схеми, що виконуються на основі транзисторних ключів. Для представлення логічних змінних в цифровій схемотехніці використовують електричне напруга, що має два різні рівні: високий, близький але рівню до напруги харчування (транзистор закритий), і низький, близький до потенціалу корпусу (транзистор відкритий). Цим рівням можна поставити у відповідність стану логічних "1" і "0". Якщо високий рівень напруги відповідає логічній "1", а низький - "0", така система позначень називається***позитивної логікою.*** В іншому випадку (високий - "0", низький - "1") система називається ***негативної логікою.***

Відповідно до трьома операціями алгебри логіки в схемі цифрових пристроїв використовують наступні логічні елементи, вхідні змінні яких часто позначають через ***x*** і, а вихідні через ***y.***

1) елемент І - схема логічного множення, кон'юнктор (рис. 3.3, ***а);***

2) елемент АБО - схема логічного додавання, діз'юнктор (рис. 3.3, ***б);***

3) елемент НЕ - схема логічного заперечення, інвертор (рис. 3.3, ***в).***

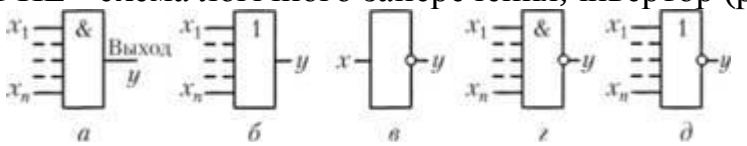


Рис. 3.3. Умовні позначення елементів цифрової логіки:
а - І; *б* - АБО; *в* - НЕ; *г* - І-НЕ; *д* - АБО-НЕ

Цей набір елементів називають основним базисом або основний функціонально повною системою елементів. Останнє означає, що за допомогою цих елементів можна створити схему, яка здійснює будь-яку як завгодно складну логічну операцію.

Крім цих елементів в інтегральній схемотехніці часто застосовуються логічні схеми, що виконують операції І-НЕ (рис. 3.3, ***г)*** і АБО-НЕ (рис. 3.3, г)).

Інформація, що надходить в цифровий пристрій, представляє дискретний (тобто складається з нулів і одиниць) сигнал (код). На передачу сигналу відводиться кінцевий відрізок часу, званий тактом роботи пристрою. Якщо за один такт в пристрій передається один з розрядів двійкового числа, то пристрій працює з послідовним кодом, якщо ж за один такт передається все двійкове число одночасно, то пристрій працює з паралельним кодом.

У загальному випадку на вхід цифрового пристрою надходить безліч двійкових змінних ***X (x*** 1; ***x*** 2, ...,***x n),*** а з виходу знімається безліч двійкових змінних ***Y (y*** 1, ***y*** 2, ... ***y*** s) • При цьому пристрій здійснює (реалізує) певний зв'язок (логічну функцію) між вхідними та вихідними змінними.

Цифрові пристрої ділять на ***комбінаційні*** і ***послідовних.***

У комбінаційних пристроях (рис. 3.4, ***а)*** значення ***Y*** протягом кожного такту визначаються значеннями ***X*** тільки в цей же такт. Такі пристрої складаються з логічних елементів. У послідовних пристроях (рис. 3.4, ***б)*** значення ***Y*** визначаються значеннями ***X*** як протягом аналізованого такту, так і що існували в ряді попередніх тактів. Для цього в послідовних пристроях крім логічних повинні бути ще й запам'ятовуючі елементи. При цьому пам'ять пристрою може охоплювати не нескінченно велике, а кінцеве число тактів. Тому цифрові (дискретні) пристрою з пам'яттю називають ***кінцевими автоматами,*** якими є всі ЕОМ.

Подібно вхідним і вихідним змінним, змінні, зберігаються в пам'яті пристрою, теж двійкові і залежать від значень вхідних змінних в попередніх тактах.

Будь-яке дискретне пристрій і складові його елементи і вузли здійснюють ту чи іншу булеву функцію над двійковими вхідними змінними.

Відомо, що булеву функцію можна задати трьома способами: ***змістовно*** (шляхом словесного опису), ***таблично*** і ***алгебраїчно.*** Найбільш часто для опису роботи дискретних пристроїв користуються табличній формою.

Таблиці, що показують зв'язок між вхідними і вихідними змінними комбінаційних пристроїв, так само як і в алгебрі логіки, називають ***таблицями істинності,*** а алгебраїчна форма цих зв'язків являє систему алгебраїчних функцій:

$$y_1 = y_1(x_1, x_2, ..., x_n),$$
$$\dots\dots\dots\dots\dots\dots$$
$$y_s = y_s(x_1, x_2, ..., x_n).$$

У послідовних пристроях вихідні змінні ***y*** і; залежать не тільки від вхідних сигналів ***x*** k, а й від сигналів елементів пам'яті, що надходять в цей же такт. При аналізі та синтезі послідовних пристрої ділять на комбінаційну частину і елементи пам'яті.

Позначимо два наступні один за одним такту роботи автомата як ***t*** і ***(t*** + 1). Стан елементів пам'яті в ***(t***+ 1) -й такт визначається множинами як вхідних сигналів, так і сигналів на виходах елементів пам'яті в попередній такт í, $z_i^{t+1} = \varphi_i(x_1, x_2, ..., x_n, I_1, I_2, ..., I_r)$ тобто.

Цей вираз називають ***функцією переходів.***

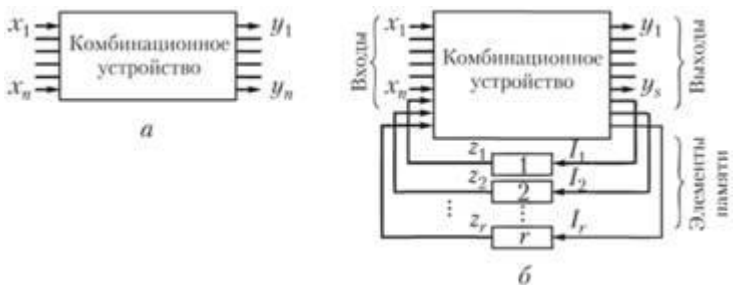


Рис. 3.4. Структура комбінаційного (а) і послідовних *(б)* цифрових пристроїв

Функції переходів і виходів послідовних пристроїв можуть виражатися ***таблицями переходів і виходів.*** Оскільки ці таблиці описують роботу послідовних пристрою, тобто процес його перемикання, вони називаються також ***таблицями перемикань.***

Реальні електричні схеми завжди інерційні, і між моментом зміни сигналів на вході схеми і моментом появи відповідного сигналу на виході завжди проходить нехай і невелике (наносекунди), але кінцевий час. Таблиці та алгебраїчні функції описують лише сталий режим, тобто в статиці. У динамічній же частині такту зв'язок між змінними може відрізнятися від режиму статики. Це явище називають перехідним станом (гонками в автоматах). Якщо його не враховувати, то спроектоване пристрій може працювати з помилками. Для боротьби з гонками в автоматах використовують синхронізацію роботи електронних схем. Відповідно до цього цифрові пристрої розділяються на асинхронні і синхронні. В асинхронних зміна вхідних сигналів відразу тягне за собою відповідну зміну вихідних сигналів (звичайно, після закінчення перехідних процесів в електронних схемах). У синхронних зміна вихідних сигналів відбувається тільки після подачі синхронізуючих (тактових) імпульсів, які керують роботою автомата.

Комбінаційні частини автоматів є асинхронними. Значить, на їх виході можуть з'являтися гонки, які призводять до збоїв (помилок), якщо елементи пам'яті автомата будуть управлятися безпосередньо вихідними сигналами комбінаційної частини. Такі автомати називають асинхронними, і в них існує небезпека збоїв.

У синхронних автоматах елементи пам'яті змінюють свій стан тільки з приходом зовнішнього тактового імпульсу, тобто коли всі перехідні процеси в комбінаційної частини закінчаться і на її виходах встановляться сигнали, відповідні вхідним. Тому небезпеки збоїв через гонок в таких автоматах немає.

ТЕМА 4. МІКРОСХЕМОТЕХНІКА.

Аналогові інтегральні мікросхеми (**АІМ**) універсальні і багатофункціональні. До таких схем належать операційні підсилювачі, інтегральні стабілізатори, компоратори та інші схеми, які складаються з базових схемотехнічних елементів, наприклад, елементарних підсилювальних каскадів, диференційних підсилювачів, каскадів зсуву потенціальних рівнів, генераторів стабільного струму, опорних елементів, кінцевих підсилювачів каскадів. Ці елементарні схеми широко використовуються як при проектуванні відомих, так і при створенні нових лінійних ІМС.**АІМ**, особливо операційним підсилювачем, властива функціональна перенасиченість за більшістю параметрів. При розробці напівпровідникових **АІС** велика увага приділяється підвищенню технологічності мікросхем, тобто зменшенню кількості технологічних операцій. Це досягається використанням транзисторних структур не тільки як елементів підсилення, а також для виконання функцій пасивних елементів, наприклад, як резисторів, конденсаторів і т.д.

Зв'язок між окремими каскадами в **АІС** звичайно безпосередній. При цьому є проблема узгодження як окремих каскадів у складі мікросхем, так і окремих каскадів між собою. Для такого узгодження необхідно, щоб потенціали вхідної та вихідної напруг були близькими до потенціалу загальної клеми джерела живлення. Цього досягають, застосовуючи каскади зсуву потенціального рівня.

Однокаскадні підсилювачі в інтегральному виконанні являють собою монолітну схему, яка містить в собі всі необхідні елементи (транзистори, діоди, резистори та інше) в інтегральному виконанні і підсилює електричні сигнали без вмикання додаткових елементів. Такі підсилювачі подібні до багатоцільових пристроїв, оскільки, змінюючи в них комутацію зовнішніх виводів, а також способи під'єднання джерела сигналів і навантаження, можна одержати підсилювачі з різними характеристиками. В окремих випадках інтегральні підсилювачі при розробці конкретних вузлів доповнюють навісними елементами.

Наведений приклад ілюструє принципово новий підхід проектування радіоелектронної апаратури з застосуванням **ІС**. В даному випадку нові вузли та блоки створюються шляхом компоновки готових функціональних вузлів. Таким чином реалізується <u>функціонально-вузловий метод проектування</u>. Параметри конкретної апаратури досягаються відповідним вмиканням виводів інтегральної схеми та розрахунком навісних елементів.

Такий самий підхід зберігається при використанні більш складних **АІС**, до яких відносяться диференційні каскади підсилення та операційні підсилювачі (**ОП**).

Диференційні підсилювачі (**ДП**) відносяться до балансних (мостових) схем підсилювачів постійного струму. Вони мають два симетричні входи і два симетричні виходи. Якість **ДП** в основному визначається ідентичністю параметрів пари транзисторів. Використовуючи стандартні схеми **ДП**, що

випускаються промисловістю, створюють різноманітні функціональні вузли (генератори гармонічних коливань, формувачі імпульсних сигналів і т.д.).

Диференційні каскади підсилення

Диференційні каскади відносять до балансних (мостових) схем підсилювачів постійного струму. Їх застосовують для зниження дрейфу нуля, що викликається зміною напруги живлення і температури нав-колишнього середовища. Диференційні каскади мають два входи і два виходи, що дозволяє проектувати інвертуючі і неінвертуючі підсилювачі і досить просто узгоджувати кола зворотних зв'язків. В диференційних каскадах легко виконати зміщення рівня вихідного потенціалу, тому можна будувати багатокаскадні підсилювачі без застосування розділяючих реактивних елементів (конденсаторів, трансформаторів). Отже, структура диференційного підсилювача узгоджена з принципами інтегральної технології, при якій можливе виготовлення пари транзисторів з майже ідентичними параметрами. При цій умові диференційні каскади мають майже ідеальні характеристики.

Диференційний підсилювач (ДП) — це балансний підсилювач по-стійного струму з джерелом стабільного струму в колі емітера. Значення цього струму обчислюють за параметрами додаткового джерела живлення і резистором в емітерному колі. На рис. показана схема ДП, яка складається з двох транзисторів і трьох резисторів. В окремих випадках напруга вхідного сигналу може бути подана лише на один вхід ($Eвх_1$=0 або $Eвх_2$= 0). Напруга вихідного сигналу j знімається або між колекторами транзисторів (симетричний вихід), або з колектора одного з транзисторів відносно заземленого провідника (несиметричний вихід).

Опір резистора R_E повинен значно перевищувати внутрішній опір підсилювача з боку його виходу, щоб значення стабільного струму $I_0 =(E-U_{BE})/R_E$ не залежало від напруги на вході ДП і було сталим навіть при наявності короткого замикання в колі навантаження джерела цього струму. Необхідно також вживати заходи для забезпечення високої стабільності струму Іо під впливом температури, оскільки параметри ДП вельми залежать від цього струму.

Важлива особливість ДП — високе підсилення різниці вхідн сигналів $Eвх_1$ -$Eвх_2$ (коли вхідні сигнали змінного струму протифазні або різнополярні для сигналів постійного струму) і значне ослаблення сумарного вхідного сигналу $Eвх_1$+$Eвх_2$. Це найчастіше сигнал завади, зумовлений напругою дрейфу нуля підсилювача.

При симетричних плечах схеми (транзистори ідентичні, а Rc_1 =Rc_2=Rc) і відсутності вхідних сигналів ДП збалансований, і напруга між колекторами (на виході) дорівнює нулю. Оскільки струм ділиться між плечами порівну, то потенціали колекторів обох транзисторів однакові і дорівнюють Uco = $Uвих_1$=$Uвих_2$ = Ec-$IoRc/2$ (рис.).

Якщо в момент часу t на вхід транзистора VTI надійшов позитивний сигнал при $Eвх_2$ = 0, то на виході лівого плеча схеми, що є підсилювальним

каскадом з ЗЕ, з'явиться підсилений сигнал $U_{вих1}$протилежної полярності (інвертований сигнал, як в схемі із ЗЕ). Одночасно на емітерному резисторі

$$R'_E = R_E h_{21B2} / (R_E + h_{21B2}),$$

де h_{11B2} — вхідний опір транзистора VT2 на емітерному вході, з'явиться позитивний імпульс $U e$, що дорівнює за амплітудою вхідному імпульсу Евх > 0 . Цей імпульс надходить на емітер транзистора VT2, викликаючи появу на колекторі підсиленого імпульсу також позитивної полярності (зміщення по фазі відсутнє, як в схемі із ЗБ) з амплітудою $U_{вих2}$ »Uвихі. Значить, вихідний сигнал з напругою $U_{вих}^2$ неінвертований по відношенню до вхідного сигналу.

Із збільшенням амплітуди вхідного сигналу $Е_{вх1}$ струм транзистора VT1 збільшується, а напруга на його колекторі знижується і, навпаки, струм транзистора VT2 зменшується, що супроводжується зростанням напругиUвих$_2$ (ділянка t_1 — t_2 на рис.). В момент часу t_2 струм транзистора VTI досягає максимального значення Io, а струм транзистора VT2 дорівнює нулю. При цьому різниця вихідних сигналів Uв*их*$_2$—*Uâèx*$_1$=IoR*c*. Описаний процес можливий, якщо між входами прикладено різницю (різнополярний) вхідних сигналів, яка називається диференційним сигналом.

При надходженні на вхід ДП синфазного вхідного сигналу *Есф*= $Е_{вх1}$ + $Е_{вх2}$ (обидва входи ДП з'єднані) і у випадку ідеального джерела струму (R_E ® ¥) сигнал на виході ДП відсутній. Оскільки в реальних ДП резистор *re* має скінчений опір, то під дією синфазного сигналу на виході підсилювача з'явиться невелика напруга розбалансування DЕсф, яка додається до корисного сигналу, зумовлюючи сигнал помилки. Тому ДП тим якісніший, чим меншу різницю вхідних сигналів він може розрізняти на фоні великого синфазного сигналу, як правило, створеного дією дестабілізуючих факторів.

4.2. Диференціальні підсилювачі; Операційні підсилювачі

В аналогових ІМС, особливо в напівпровідникових, між окремими підсилювальними каскадами відсутні роздільні конденсатори, тому для з’єднання окремих каскадів підсилення використовується лише гальванічний (омічний) зв’язок (зв’язок за постійним струмом). Таким чином, розширюється межа частотної характеристики підсилювача до $f_н$ = 0. Тобто АІС є підсилювачами постійного струму. В той же час омічний зв’язок між каскадами обумовлює необхідність додаткових схемних рішень. Найбільш вагомими є задачі стабілізації режиму підсилювального каскаду та зсуву рівнів напруг.

4.2.1. Схеми стабілізації режиму роботи каскаду підсилення.

Нестабільність положення робочої точки інтегрального транзистора може бути спричинена старінням елементів ІМС, дрейфом параметрів мікросхеми, нестабільністю напруги джерела живлення. Через відсутність роздільних конденсаторів повільна зміна напруги підсилюється наступними каскадами, а в деяких випадках, коли інформаційний сигнал має постійну складову, небажані зміни напруги внаслідок дрейфу викликають значні похибки. Тому стабільність

точки спокою є запорукою якісної роботи АІС. Найбільший вплив на зміщення робочої точки чинить температурна нестабільність параметрів елементів ІМС.

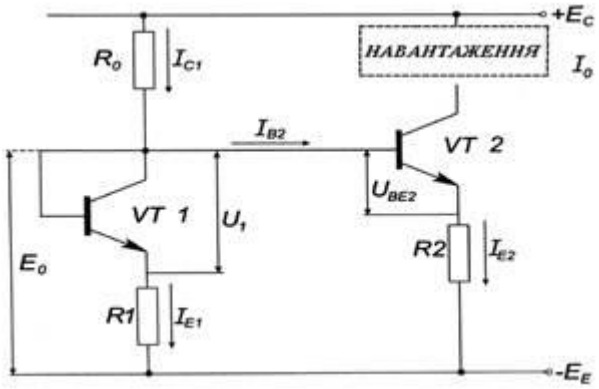
У дискретних схемах та у гібридних ІМС для забезпечення режиму спокою і його термостабілізації використовують резистивні кола зміщення і введення місцевих кіл негативного зворотного зв'язку. При цьому резистори шунтуються конденсаторами великої ємності, що виключає вплив цих кіл на передачу інформаційного сигналу.

У напівпровідникових ІМС застосуванням елементів негативного зворотного зв'язку є недоцільним, оскільки формування конденсаторів великих ємностей практично неможливе.

Під впливом зміни температури зміщуються статичні характеристики транзистора (див. рис.4.11). З підвищенням температури змінюється колекторний (вихідний) струм I_C.

Таким чином, для стабілизації режиму АІС у схемі необхідно передбачати генератор стабільного струму, який забезпечить незмінний струм в навантаженні при зміні його опору або вхідної напруги. Нагадаємо, що генератор струму характеризується великим внутрішнім опором, який значно перевищує опір навантаження ($R_г >> R_н$).

У напівпровідникових ІМС формування резисторів з великим опором недоцільне (планарні резистори займають велику площу). Тому використовують *параметричні методи* температурної стабілізації положення робочої точки.



Показані напруги та струми зв'язані таким рівнянням:

$\mathbf{E_0 = U_1 + I_{E1}R1 = U_{BE2} + R_2 I_{E2}}$.

Знехтуємо малим струмом $\boldsymbol{I_{B2}}$, тоді $\boldsymbol{I_{E1} = I_{C1}}$ і $\boldsymbol{I_{E2} = I_0}$.

Якщо опори резисторів $\boldsymbol{R_1 = R_2}$ однакові, а параметри та характеристики транзисторів VT_1 та VT_2 співпадають, що досягається в ІМС, то $I_0 = I_1$. Тобто струм у навантаженні повторює вхідний струм I_{C1}. Таку схему називають "***струмовим дзеркалом***".

Відносна нестабільність струмів транзисторів VT1 і VT2 однакова.

Таким чином, щоб стабілізувати струм I_0 (I_{C2}), необхідно з достатньою точністю стабілізувати струм I_{C1} .

Вхідний струм $\boldsymbol{I_{C1} = (E_C - E_0)/R_0}$. За умови $\boldsymbol{E_C >> E_0}$ струм $\boldsymbol{I_{C1}}$визначається зовнішніми параметрами E_C та R_0. Отже, заданий режим транзистора $VT1$ і його колекторний струм I_{C1} (а відтак і струм I_0) можна забезпечити добиранням зовнішніх елементів: резистора R_0 і напруги джерела живлення. Визначають їх з урахуванням допустимих значень відносної зміни температури $\Delta T/T$, опору резистора $\boldsymbol{\Delta R_0/R_0}$, струму навантаження $\Delta I_0/I_0$ та напруги джерела живленняE_C/E_C .

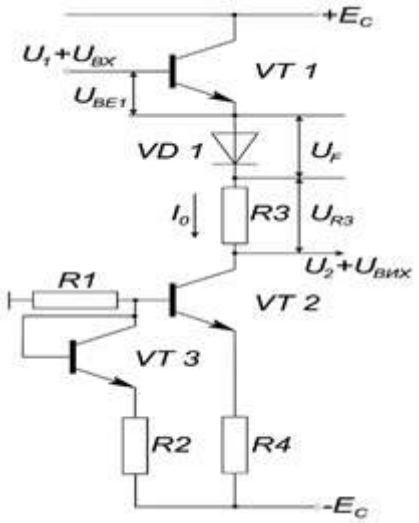
Якщо $R_1 \neq R_2$, **то і** $I_{E1} \neq I_{E2}$, але зберігається рівняння $U_1 = U_{BE2}$ **і** $I_{E1}R_1 = I_{E2}R_2$. Звідки $I_0 = I_1(R_1R_2)$.

У даному випадку струм може наслідувати струм I_{C1} як в "збільшеному" так і в "зменшеному" масштабі залежно від співвідношення опорів. Цей масштаб не перевищує декількох одиниць.

Розглянута схема широко використовується для стабілізації режиму диференціальних підсилювачів як в окремому виконанні так і в складі операційних підсилювачів.

4.2.2. Схеми зсуву рівнів напруг

Інтегральні підсилювачі є підсилювачами постійного струму. Завдяки омічному зв'язку на базу транзистора кожного наступного каскаду поступає не тільки корисний інформаційний сигнал, але й постійна складова напруги з колектора попереднього каскаду.



В багатокаскадних підсилювачах ця складова "накопичується", що викликає певні затруднення. Дійсно, напруга U_{BE} = 1В. Напруга на колекторі попереднього каскаду в точці спокою може досягати значень $0{,}1E_C \ldots 0{,}5E_C \ldots E_C$, тобто значно перевищує необхідну напругу U_{BE}. Щоб забезпечити необхідний режим в даному випадку, підвищують потенціал емітера. Це досягається вмиканням в його коло резистора, що зменшує коефіцієнт передачі інформаційного сигналу.

Для того, щоб усунути постійну складову на вході наступного каскаду, але по можливості без втрат передати корисний сигнал, використовують спеціальні схеми зсуву рівнів напруг. Такі схеми забезпечують стабільну роботу каскадів, не вносячи помилок в постійну складову сигналу зі зміною напруги живлення і температури навколишнього середовища. Найчастіше схеми зміщення рівнів будують на основі генераторів стабільного струму.

Найпростішою схемою зсуву рівнів напруг є емиітерний повторювач. Він є основою інших більш складних схем.

Розглянемо процес зменшення постійної складової напруги U_1, що діє на базі VT1 разом зі змінним інформаційним сигналом U_{BX}. Потенціал емітера VT1 нижчий від потенціалу бази на незначну величину U_{BE1}. Коефіцієнт передачі за напругою G_U = 1.

На діоді VD1 спадає пряма напруга $U_F = U_{BE1}$. За необхідності включають *m* діодів.

Транзистори VT2, VT3, резистори *R*1, *R*2 та *R*4 утворюють генератор стабільного струму I_0, який забезпечує необхідний спад напруги на резисторі *R*3.

Співвідношення між вхідним U_1 та вихід-ним U_2 постійними рівнями

$$U_1 - U_2 = (m+1)\, U_F + I_0R_3$$

Змінюючи значення m, I_0 та $R3$ можна забезпечити будь-який зсув рівня напруги. При цьому змінний інформаційний сигнал передається майже без спотворень.

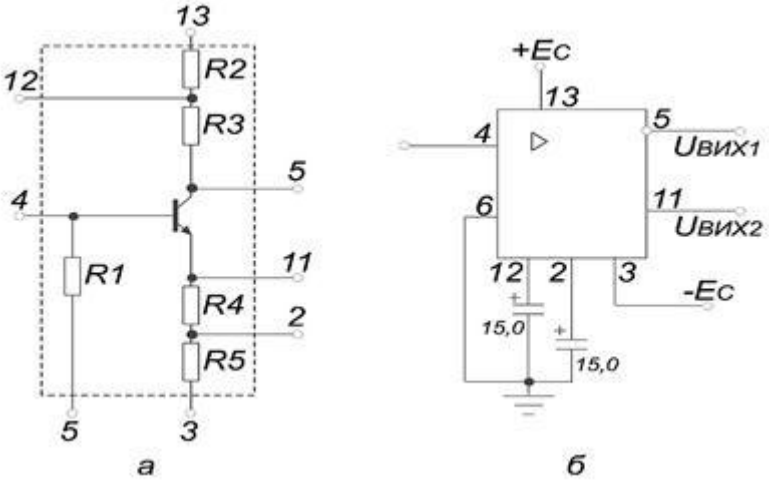
Такі схеми широко використовують у багатокаскадних інтегральних підсилювачах, зокрема в операційних підсилювачах.

4.2.3.Однокаскадні багатоцільові підсилювачі

Однокаскадні підсилювачі в інтегральному виконанні являють собою монолітну схему, яка містить в собі всі необхідні елементи (транзистори, діоди, резистори і ін.) в інтегральному виконанні та підсилює електричні сигнали без вмикання додаткових елементів. Такі підсилювачі подібні до багатоцільових пристроїв, оскільки, змінюючи в них комутацію зовнішніх виводів та способи приєднання джерела сигналів і навантаження, можна одержати підсилювачі з різними характеристиками та різними схемами вмикання (СЕ, СБ, СК). В окремих випадках інтегральні підсилювачі при розробці конкретних вузлів доповнюють навісними елементами.

При такому вмиканні вхідний інформаційний сигнал подається на вхід *4*, а знімається з виходів *5* та *11*. На виводі *5*, як і в схемі із ЗЕ сигнал змінює полярність. Це інвертуючий вихід, що відповідно позначається на схемі вмикання. Такеж позначення використовують і в цифрових ІМС для виділення інвертуючих виводів.

Опором навантаження колекторного кола змінному струму є резистор $R3$, тому що приєднаний до виводу *12* і корпусу навісний конденсатор ємністю 15,0 мкФ шунтує за змінним струмом резистор R_2. Таке вмикання дозволяє забезпечити термостабілізацію робочої точки дією негативного зворотного зв'язку за напругою для постійного струму (колекторна стабілізація). Зміна температури, а отже і струму колектора відбувається повільно. Конденсатор не шунтує резистор $R2$ і на ньому формується додатковий спад напруги, викликаний збільшенням колекторного струму при підвищенні температури. Це - схемне вмикання, яке протидіє зміні колекторного струму. Так само резистор $R5$ з конденсатором великої ємності 15,0 мкФ утворює ланцюжок емітерної термостабілізації внаслідок негативного зворотного зв'язку за постійним струмом.



На виході *11* формується неінвертований сигнал, як і в схемі із СК. При вмиканні АІС, показаному на рис.6.4, *б,* вихідні сигнали формуються одночасно на двох виходах *5* і *11.* Тобто створена можливість одержувати протифазні (парофазні) сигнали.

Відповідним вмиканням зовнішніх навісних конденсаторів можна забезпечити побудову підсилювача із СЕ, СБ або із СК.

Наведений приклад ілюструє принципово новий підхід проектування радіоелектронної апаратури із застосуванням ІМС. У даному випадку нові вузли та блоки утворюються за допомогою готових функціональних вузлів. Таким чином, реалізується функціокально- вузловий метод проектування Параметри конкретної апаратури досягаються відповідним вмиканням виводів ІМС та розрахунком навісних компонентів.

Такий самий підхід зберігається при використанні більш складних АІС.

4.2.4.Диференціальні підсилювачі

Диференціальні підсилювачі (ДП) належать до балансних (мостових) схем підсилювачів постійного струму. Якість ДП здебільшого визначається ідентичністю параметрів пари транзисторів. У дискретній транзисторній схемотехніці це виконати важко, а тому такі схеми використовують рідко. Структура диференціального підсилювача узгоджена з принципами інтегральної технології, за якою можливе виготовлення пари транзисторів з майже ідентичними параметрами (дві структури розташовані поруч на одній підкладці і формуються одночасно за однакових умов).

Схеми ДП були розроблені і широко використовувалися ще на етапі електровакуумної електроніки. Їх впровадження було одним з ефективних напрямів побудови підсилювачів постійного струму (ППС), які дозволяли обробляти інформаційні сигнали в пристроях автоматичного контролю та регулювання для регістрації таких величин, як потужність, кут зсуву фаз, тиск, температура, світловий потік, прозорість і ін. Такі електричні та неелектричні параметри зазвичай просто перетворюються в струми або напруги, що повільно змінюються, а відтак частота таких сигналів складає тільки одиниці або навіть частки герца. Для підсилення таких струмів або напруг необхідні підсилювачі, смуга частот яких має нижню межу $f_н = 0$.

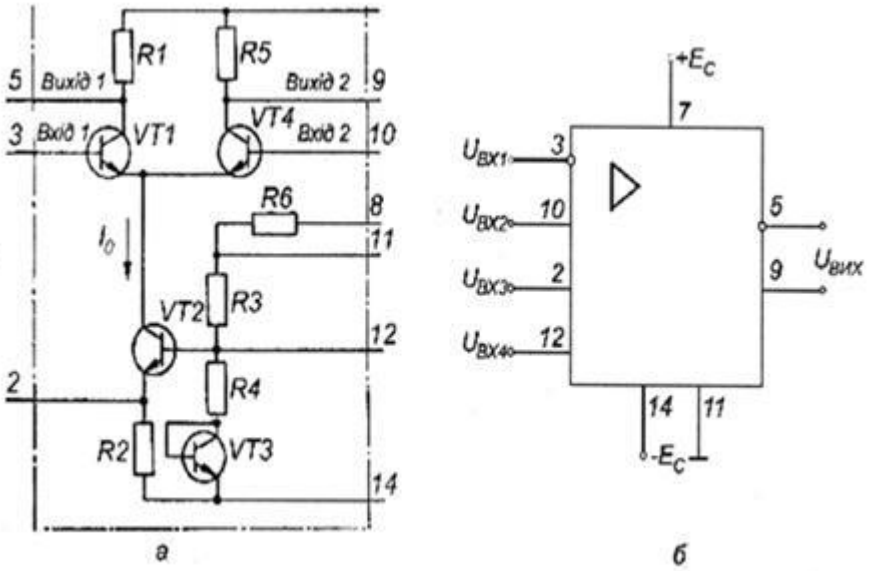
Підсилювачі постійного струму широко використовуються в електронних обчислювальних машинах і вимірювальній техніці, медицині, ядерній фізиці та в інших областях техніки.

При побудові та експлуатації ППС виникає важлива задача: забезпечення стабільної роботи підсилювача при зміні напруги джерела живлення, режимів транзистора, параметрів елементів та інших дестабілізуючих факторів. Формування вихідної напруги під дією цих факторів фактично не відрізняється від змін, викликаних дією на вході корисних сигналів.

Зміни вихідної напруги, що не пов’язані зі вхідними інформаційним сигналом, а зумовлені внутрішніми процесами в підсилювачі, називають *дрейфом нуля* підсилювача. Напруга дрейфу на виході ППС може виявитись одного порядку з напругою корисного сигнала. Це викликає недопустимі інформаційні спотворення.

Схеми диференціальних підсилювачів будуються за мостовими схемами, що дозволяє вирішити описану вище проблему через суттєве зменшення дрейфу нуля.

Типова принципова схема та схема вмикання диференціального підсилювача в інтегральному виконанні показана на рис. Це мікросхема К118УД1: напівпровідниковий однокаскадний диференцальний підсилювач постійного струму серії 118. По суті така схема являє собою міст, плечами якого є резистори $R1 = R5$ і внутрішні опори транзисторів VT1 та VT4 (диференціальна пара).



До однієї діагоналі моста (виводи *7* і *14*) підводиться напруга одного або двох джерел живлення ($+E_C$ і $-E_E$), а до другої (колектори VT1 і VT4) вмикається навантаження (виводи *5* і *9*). Якість роботи такої схеми визначається симетрією обох пліч, тобто ідентичністю параметрів транзисторів VT1 та VT4, рівністю опорів *R*1 і *R*5. У початковому стані (до появи інформаційного сигналу) ДП повинен бути збалансований, а значить напруга на навантаженні дорівнює нулю ($U_н = |U_{C1} - U_{C2}| = 0$). При дії різних дестабілізуючих факторів (наприклад, підвищенні $+E_C$) одночасно зміниться напруга колекторів, а різницева напруга колекторів, тобто напруга на навантаженні, залишиться незмінною ($U_н \approx 0$).

Для стабілізації режиму роботи в схемі використовується генератор стабільного струму на транзисторі VT2, функціонування якого описано в розділі 6.3.2. Коло зміщення побудоване на резисторах *R*3, *R*4, *R*6 та транзисторі VT3 у діодному вмиканні. Це забезпечує режим роботи генератора стабільного струму і температурну стабілізацію мікросхеми.

При вмиканні навантаження між колекторами транзиторів VT1 і VT4 використовують *симетричний вихід*, а при зніманні напруги вихідного сигналу з колектора одного з транзистрорів відносно заземленого провідника – *несиметричний вихід*. Вихідний інформаційний сигнал може подаватися: на один із входів (*3* або *10*) відносно заземленої точки (*несиметричний вхід*), або ж одночасно на два входи (*симетричний вхід*).

При подачі інформаційного сигналу на симетричні входи на базах вхідних транзисторів VT1 і VT4 діють напруги протилежних знаків ($\Delta U_{B1} = -\Delta U_{B2}$). Такі сигнали називають диференціальними. Ідеальний ДП реагує тільки на диференціальний сигнал, звідси – назва цього типу підсилювачів.

Ефективно проявляється важлива особливість диференціального підсилювача, яка обумовила їх широке використання. Це: значне підсилення інформаційного (диференціального) сигналу та *значне ослаблення синфазного сигналу завади.*. Синфазні вхідні напруги U_{Cl} – це напруги між кожним з входів ІМС та спільним виводом, амплітуди та фази яких збігаються Коли джерело інформаційних сигналів (датчик) вмикається до симетричного входу диференціального підсилювача, на ньому діють протифазні або різнополярні сигнали.

Амплітуда вихідного сигналу визначається різницею напруг на симетричному вході. У результаті відповідно змінюється напруга на колекторах транзисторів VT1 і VT4. Коефіцієнт підсилення диференціального сигналу

A_U визначається відношенням приросту вихідної напруги до приросту вхідної за формулою:

$$A_U = \frac{|U_{C1} - U_{C2}|}{|U_{BX1} - U_{BX2}|}$$

Напруга завади діє синфазно, тобто або підвищує, або зменшує одночасно напругу на обох входах *3* і *10*. Це викликає відповідні синфазні зміни напруг на колекторах. У випадку ідеального джерела струму сигнал на виході диференціального підсилювача відсутній. У реальних схемах під дією синфазного сигналу на виході існує синфазна напруга розбалансування $\Delta E_{сф}$, яка додається до корисного сигналу, зумовлюючи сигнал помилки. Коефіцієнт підсилення синфазного сигналу A_{UC} значно менший за A_U. Диференціальний підсилювач тим якісніший, чим меншу різницю вихідних сигналів він може розрізнити на фоні великого синфазного сигналу, зазвичай створеного дією дестабілізуючих факторів. Ця важлива властивість ДП оцінюється *коефіцієнтом ослаблення синфазної складової* (в децибелах):

$$K_{CMR} = 20\lg\frac{|A_U|}{|A_{CC}|}$$

Типові значення цього параметра - 60…80 дБ. Це свідчить про високу завадостійкість підсилювача. Мікросхема КІІ8УДІА забезпечує ослаблення вхідних синфазних напруг не менше ніж 60 дБ ($G_{A1}/G_{сф} = 10^3$).

Диференціальні підсилювачі належать до типових схем універсального призначення. Вхідні сигнали можна подавати не лише на диференціальні входи *3* і *10*, але і на входи генератора постійного струму *2* і *12*. Це також може бути або симетричний, або несиметричний вхід.

Багато входів і виходів забезпечують широкі можливості введення і комбінування негативного та позитивного зворотного зв`язків для одержання якісних показників диференціального підсилювача та побудови різних функціональних пристроїв.

Висока завадостійкість і термостабільність диференціальних підсилювачів зумовили їх широке використання як вхідних каскадів, зокрема в операційних підсилювачах.

У технічній літературі із схемотехники, в науково-технічних журналах, довідниках представлено великий вибір диференціальних підсилювачів зі схемами вмикання та параметрами зовнішніх елементів для побудови підсилювачів, генераторів, компараторів, змішувачів частоти і ін. Вони можуть бути ефективно використані в курсовому, дипломному проектуванні та в практичнів інженерній діяльності для реалізації сучасного функціонально-вузлового методу проектування.

4.2.5. Операційні підсилювачі

Основне призначення ОП – побудова за допомогою зовнішніх навісних елементів схем із фіксованим коефіцієнтом підсилення і точно синтезованою передавальною функцією. Їх використовують для побудови широкої гами різновидів пристроїв. Стандартний підсилювач загального призначення може використовуватись приблизно в 100…150 схемах вмикання.

Схемотехніка ОП була відома ще до появи лінійних ІМС. У класичній електроніці до класу ОП відносили багатокаскадні підсилювачі постійного струму зі зворотним зв’язком. Їх використовували в аналоговій обчислювальній техніці для виконання операцій алгебраїчного додавання, віднімання, множення, ділення,

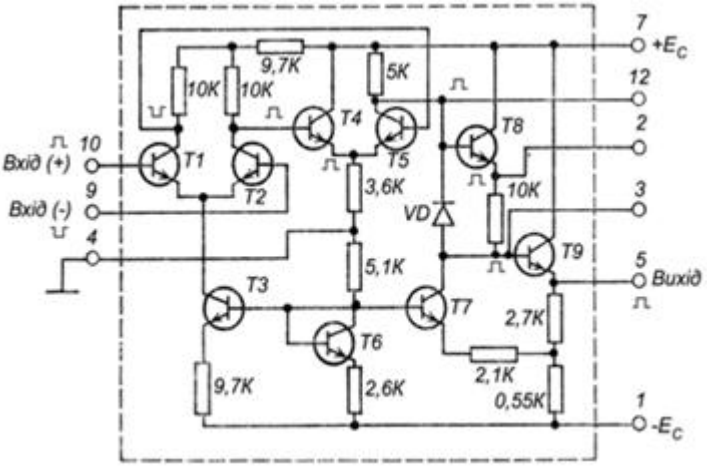
диференціювання, логарифмування тощо. Це і зумовило їхню назву - операційні (розв'язувальні) підсилювачі.

В інтегральній схемотехніці ОП - це підсилювач постійного струму, який характеризується великим вхідним, низьким вихідним опорами і дуже високим коефіцієнтом підсилення за напругою. Якщо уявити ОП ідеальною моделлю, то він повинен мати такі властивості: $R_{ВХ}$ ®¥, $R_{ВИХ}$ ®0 і G_U ®¥. Успіхи планарної технології зумовили появу серійних партій ОП у вигляді інтегральної мікросхеми, що дозволило значно удосконалити їхні технічні й експлуатаційні показники. Такі ОП тепер використовуються не лише для виконання математичних операцій, а й для підсилення, перетворення, формування і обробки електричних сигналів.

Зазвичай ОП складаєтся з трьох каскадів: вхідного, побудованого, як правило, за схемою диференціального підсилювача, проміжного підсилювача напруги та вихідного каскаду, побудованого за схемою із ЗК з додатковими елементами для покращання параметрів ОП.

Принцип побудови та функціювання складових елементів ОП розглянемо на прикладі найпростішої схеми, яка використовувалась в перших зразках інтегральних ОП. Зокрема на прикладі інтегральної мікросхеми 140УД1 (А, Б) – напівпровідникової ІМС серії 140. Конструктивно ОП 140УД1 розміщується на кремнієвій пластині розміром 1,1х1,1 мм.

Як вхідний каскад використовується один з видів диференціального підсилювача на транзисторах VT1 та VT2 з джерелом стабільного струму на транзисторі VT3 і колом температурної стабілізації на транзисторі VT6 в діодному ввімкненні (струмовим дзеркалом).



Другий (проміжний) каскад ОП виконаний на транзисторах VT4 та VT5 також за схемою диференціального підсилювача, але з несиметрисним виходом. Він виконує дві функції: підсилення напруги та перетворення двофазного сигналу, що формується на колекторах транзисторів VT1 та VT2 в однофазний, що формується на колекторі транзистора VT5.

Струм другого каскаду не фіксується джерелом стабільного струму в колі емітерів транзисторів, оскільки на його входи поступає вже підсилений інформаційний сигнал, в якому синфазна складова практично відсутня (заглушається першим каскадом). Це дозволяє реалізувати в другому каскаді режим роботи з міліамперними струмами, що забезпечує підсилення напруги з коефіцієнтом порядку кількох сотень.

При використанні диференціальних підсилювачів максимальний коефіцієнт підсилення за напругою одержують при підключенні навантаження до симетричного виходу. Операційні підсилювачі мають однофазний вихід. Щоб забезпечити перехід від симетричного виходу до несиметричного без втрати

коефіцієнта підсилення за напругою транзистор VT4 вмикають за схемою із СК. Розглянемо як відбувається передача сигналів.

Припустимо, що на симетричні входи ОП подається імпульсний сигнал, на вході *10* напруга збільшується, на вході *9* – зменшується. На рис.6.8 це зображається позитивним та негативним імпульсами. У результаті на колекторах транзисторів VT1 та VT4 сформуються інвертовані імпульси. З колектора транзистора VT2 імпульс позитивної полярності подається на базу транзистора VT4, ввімкнутого за схемою із СК (емітерний повторювач). На об'єднаних емітерах транзисторів VT4 та VT5 формується позитивний імпульс. Потенціал емітера транзистора VT5 зростає, а напруга U_{BE} цього транзистора зменшується, він закривається, струм колектора зменшується, зумовлюючи підвищення напруги на колекторі ($U_C = E_C - I_C R_C$). Одночасно негативний імпульс з колектора транзистора VT1 подається на базу транзистора VT5, що також спричиняє зменшення колекторного струму. Таким чином, на колекторі транзистора VT5 формується однофазний сигнал, амплітуда якого пропорційна обом сигналам симетричного виходу першого каскаду.

Вихідний каскад ОП, виконаний на транзисторах VT7...VT9, є однотактним підсилювачем, що працює в режимі підсилення А. Транзистор VT8, увімкнений за схемою емітерного повторювача, забезпечує зсув рівня напруг. За відсутності вхідних сигналів і використанні двох джерел живлення напруга на виході (на емітері транзистора VT9) повинна дорівнювати нулю, а на базі – близько 0,4…0,8 В. Напруга на колекторі транзистора VT5 значно перевищує цю величину (може досягати $+E_C$). Повторювач на транзисторі VT8 узгоджує проміжний та вихідний каскади за рівнем постійної напруги. Зарядна ємність діода VD1 діє як прискорювальний конденсатор, зменшуючи спотворення різких перепадів сигналу.

Транзистор VT7 виконує дві функції: є генератором стабільного струму в схемі зсуву постійного рівня напруги, забезпечуючи необхідний спад напруги на резисторі в колі емітера транзистора VT8; вмикається в коло позитивного зворотного зв'язку вихідного каскаду – емітерного повторювача на транзисторі VT9. Це дозволяє підвищити вхідний опір всього вихідного каскаду, зменшити вихідний опір та підвищити коефіцієнт підсилення до 5. Підсилювач 140УД1 забезпечує коефіцієнт підсилення за напругою в декілька тисяч в діапазоні частот до 5 МГц.

Таким чином, позитивний імпульс, що формується на колекторі транзистора VT5, підсилюється і без зміни полярності виділяється на виході *5*. Сигнал такої полярності подавався на вхід *10*, тому цей вхід називають *неінвертувальним*. На вхід *9* подавали сигнал протилежної полярності (відносно сигналу, одержаного на виході *5*), тому цей вхід називають*інвертувальним*. На умовному графічному позначення ОП такий вхід позначають кільцем.

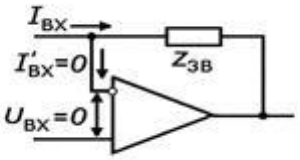
Залежно від схеми вмикання ОП до джерела інформаційних сигналів розрізняють: диференціальне, інвертувальне і неінвертувальне вмикання ОП. Для спрощення розрахунків і аналізу схем користуються згаданою вище ідеальною моделью. Оскільки $R_{вх} = \infty$ вихідний струм ОП $I_{вх} = 0$. Через нескінченно великого внутрішнього коефіцієнта підсилення і вихідного опору, що дорівнює нулю, максимальна амплітуда вихідного сигналу, яка не може перевищувати напругу джерела живлення (як і в звичайних підсилювачах), досягається при вхідній напрузі майже дорівнює нулю адже $G_U = \infty$). Все це відповідає принципу віртуального замикання вхідних зажимів ОП (рис.6.9). При віртуальному замиканні, як і при звичайному замиканні, напруга між замкненими затискачами дорівнює нулю. Але в даному випадку струм між віртуально замкненими затискачами не протікає, тобто для струму віртуальне замикання еквівалентне розриву кола. З урахуванням цього проведемо аналіз згаданих вище схем вмикання ОП.

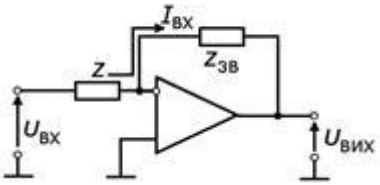
4.2.6.Інвертувальна схема вмикання ОП

Така схема показана нарис.6.10. Використовуючи принцип віртуального замикання, знаходимо, що $I_{вх} = (U_{вх} - 0)/Z$, а вихідна напруга $U_{вих} = -I_{вх}Z_{зв} - 0$. Звідки коефіцієнт передачі напруги:

$$G_U = U_{вих}/U_{вх} = -Z_{зв}/Z .$$

Таким чином, коефіцієнт передачі напруги ОП зі зворотним зв'язком визначається тільки співвідношенням зовнішніх елементів зворотного зв'язку. Але треба мати на увазі, що помилка такого визначення тим менша, чим ближче параметри ОП наближаються до ідеальної моделі. Загальний коефіцієнт підсилення типових ОП може досягати 100000 і більше, що дозволяє користуватися одержаним співвідношенням.





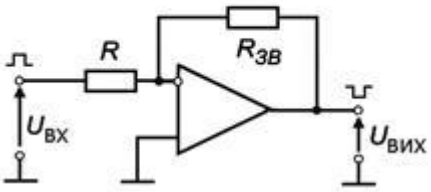
Вибираючи відповідний тип опору (резистор, конденсатор, індуктивність або їх комбінацію) одержують необхідний функціональний вузол.

Для побудови *масштабного пілсилювача* використовують резистори.

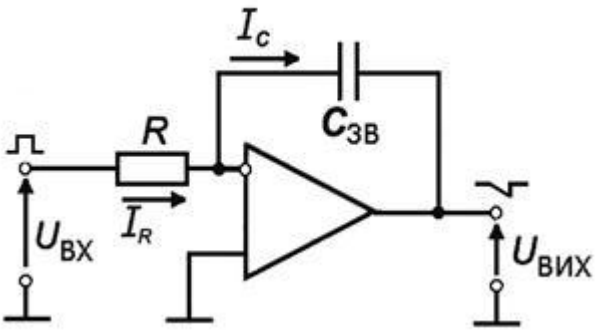
Вихідна напруга

$$U_{вих} = -(R_{зв}/R_1)U_{вх} .$$

Тобто відбувається зміна масштабу електричної величини множенням вхідного сигналу на деякий сталий коефіцієнт ($R_{зв}/R_1$).



Інвертуальне вмикання ширико використовують для побудови суматорів. Для цього до входу підключають декілька джерел інформаційних сигналів. Завдяки різним співвідношенням опорів резисторів встановлюється необхідний ваговий коефіцієнт,який визначає величину відповідної складової у формуванні вихідного сигналу.



В інтеграторі напруги, активний опір зворотного зв'язку замінений реактивним елементом – конденсатором Враховуючи параметри ідеальної моделі ОП, струм, що протікає через резистор R, знаходять за формулою: $I_R = U_{вх}/R = I_C$

Напругу на виході визначають напругою на конденсаторі::

$$U_{вих} = -U_C = -\frac{1}{C}\int I_C dt = -\frac{1}{C}\int I_R dt = -\frac{1}{RC}\int U_{вх} dt$$

Якщо до входу ОП прикласти напругу у вигляді стрибка із сталою амплітудою $U_{вх}$, то

$U_{вих} = -U_{вх}t/RC$,

де RC - стала часу інтегратора.

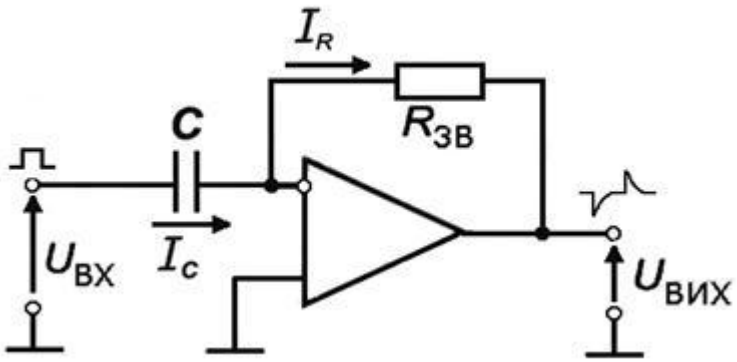
Інвертувальний підсилювач реагує на низькочастотні вхідні інформаційні сигнали аж до частоти $f_н = 0$, тобто на сигнали постійного струму. Амплітуда і форма вихідного сигналу залежать від амплітуди вхідного сигналу $U_{вх}$ та співвідношення тривалості вхідного імпульса і сталої часу інтегратора. Тому інтегратор можна ефективно використовувати як низькочастотний функціональний вузол оптимальної обробки сигналів, зокрема для побудови генераторів пилоподібної напруги. Якість виконання операції інтегрування тим вища, чим більша стала часу порівняно з тривалістю вхідного сигналу.

В такому присторі елемент Z інвертувальної схеми вмикання являє собою реактивний компонент – конденсатор. Враховуючи особливості ідеальної моделі ОП, струм, що протікає через конденсатор, можна визначити за формулою

$I_C = C(dU_{вх}/dt) = I_R$.

Тоді напруга на виході

$U_{вих} = -I_R R = -RC(dU_{вх}/dt)$,



де $RC = t$- стала часу.

Точніть виконання операції диференціювання визначається співвідношенням тривалості вхідного сигналу t_c та сталої часу.

Якщо $t_c >> t$, то при подачі на вхід прямокутного імпульсу тривалістю t_c на виході одержимо продиференційований сигнал (дуже короткий негативний та

позитивний стрибки напруги, (рис.6.13). Полярність імпульсів показана з урахуванням інвертувального вмикання ОП.

4.2.7.Неінвертувальне вмикання.

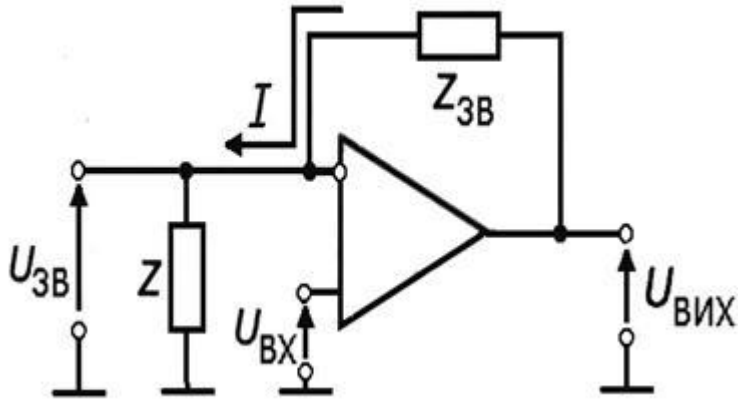
Напруга зворотного зв'язку з виходу подається на інвертувальний вхід підсилювача, а вхідний сигнал - на неінвертувальний вхід. За умов ідеального ОП коефіцієнт передачі за напругою визначають спів- відношення між вихідною напругою $U_{вих}$ і напругою зворотного зв'язку $U_{з.з}$, яка формується на опорі Z:

$$U_{з.з} = IZ = \frac{U_{вих}}{Z_{з.з} + Z} Z$$

Враховуючи принцип віртуального замикання (рис.6.7), одержують $U_{зв} = U_{вх}$, а відтак:

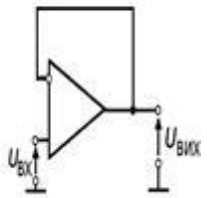
$$G_U = \frac{U_{вих}}{U_{вх}} = \frac{Z_{зз}}{Z} + 1$$

При неінвертувальному вмиканні ОП коефіцієнт передачі за напругою так само як і при інвертувальному, визначають зовнішніми навісними елементами.



Якщо $Z_{з.з}$ = 0, то G_U = 1. Це дозволяє побудувати повторювач напруги $U_{вих} = U_{вх}$. Вихідна напруга повторює вхідну за фазою і амплітудою. Схема повторювача показана на рис.6.13. Вона відрізняється дуже високим вхідним опором та малим вихідним опором. Останнє дозволяє підключати до ОП навантаження з опором 0,3…1 кОм.

Повторювачі напруг широко застосовуються як вхідні та вихідні вузли радіоелектронних пристроїв.



Промисловість випускає багато типів ОП.

Операційні підсилювачі 140УД6; 140УД14 побудовані за двокаскадними схемами з використанням супербета транзисторів. Завдяки таким транзисторам ці ОП мають вхідний опір $2\cdot10^3$ і $30\cdot10^3$ Ом при дуже малих струмах (30 і 2 мкА) відповідно. Коефіцієнт підсилення згаданих ОП $G_U = (50…70)\cdot10^3$. У схемі ОП типу 544УД1 вхідний опір підвищений до 100 МОм завдяки використанню у вхідному диференціальному каскаді ПТ.

Розроблені також інтегральні прецизійні ОП з високою стабільністю характеристик, малими шумами і низьким рівнем дрейфу нуля. До таких схем належить ОП 153УД5.

4.2.8. Імпульсний режим ОП

Для використання ОП в аналогових пристроях (підсилювачі, генератори гармонічних коливань та ін.) вхідний сигнал повинен мати такий рівень, щоб використовувати нахилені ділянки кривих передавальної характеристики, коли вихідна напруга пропорційно залежить від вхідної. В імпульсному режимі роботи рівні вхідного сигналу перевищують значення, які відповідають лінійній області передавальної характеристики. При цьому вихідна напруга досягає $U^+_{вих.max}$ або $U^-_{вих.max}$. Ці рівні вихідної напруги є сталими (сприятлива умова для формування незмінної вершини імпульсу) і майже дорівнюють напругам джерел живлення $E_c^{\pm}$

та E^{-}_{C} ОП. Незмінність означених рівнів за величиною зумовлено незмінністю за величиною горизонтальних ділянок кривих передавальної характеристики ОП, які відповідають режиму повністю закритого (режим відсікання) або повністю відкритого (режим насичення) транзистора вихідного каскаду (найчастіше емітерний повторювач) ОП.

Таким чином, ОП, як і поодинокий біполярний транзистор, може працювати як у лінійному, так і в імпульсному режимі.

Властивості та параметри ОП дозволяють ефективно використовувати їх для побудови імпульсних пристроїв, зокрема компараторів. У таких функціональних вузлах порівнюються дві напруги, що надходять до входів (або на один вхід) підсилювача – вхідна, що змінюється, і опорна. Опорна напруга позитивної або негативної полярностей незмінна за величиною. Коли вхідна напруга досягає рівня опорної, на виході ОП відбувається зміна полярності напруги, наприклад, $U^{-}_{вих.max}$ на $U^{+}_{вих.max}$.

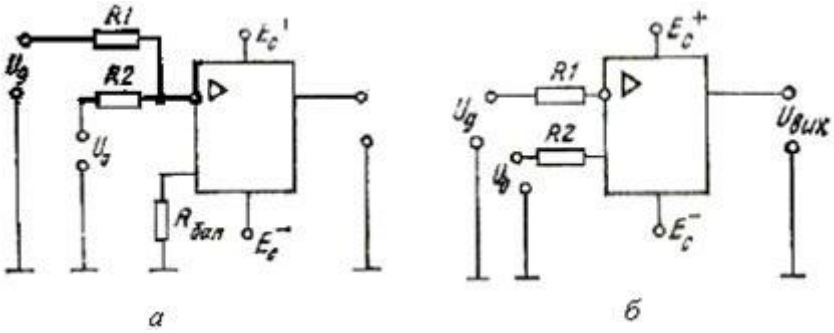
Компаратори напруги. Компараторне ввімкнення ОП використовують для порівняння напруги джерела сигналів $U_д$ з опорним сигналом U_0. Компараторний режим ОП частіше застосовують без зовнішніх кіл негативного зворотного зв'язку, подаючи порівнювані сигнали на один або обидва входи підсилювача.

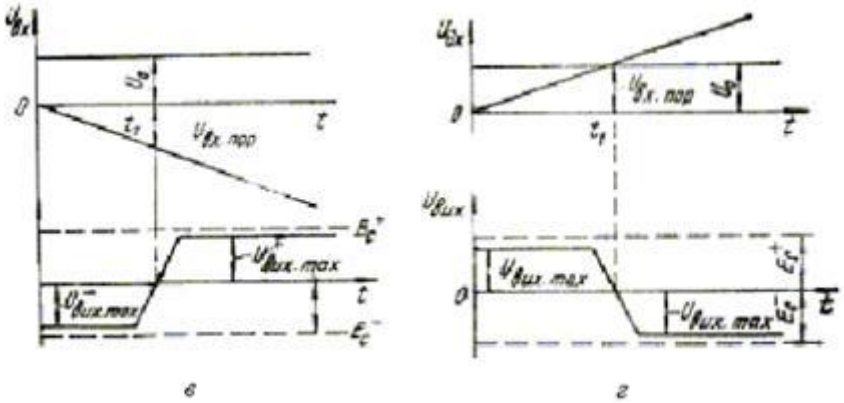
Для порівняння різнополярних напруг на вході використовують одновходовий компаратор, в якому опорний сигнал і досліджуваний надходять до інвертувального входу ОП. В інтервалі часу 0 ... t_1 виконується нерівність $|U_д| < |U_0|$, тому $U_{вх} > 0$ і напруга на виході компаратора $U_{вих} = U^{-}_{вих.max} \approx E^{-}_{C}$ (напруга на інвертувальному вході і на виході різнополярні). У момент часу t_1 вхідний сигнал досягає порогового значення

$U_I = U_{вх.пор} = U_0 R_1 | R_2$

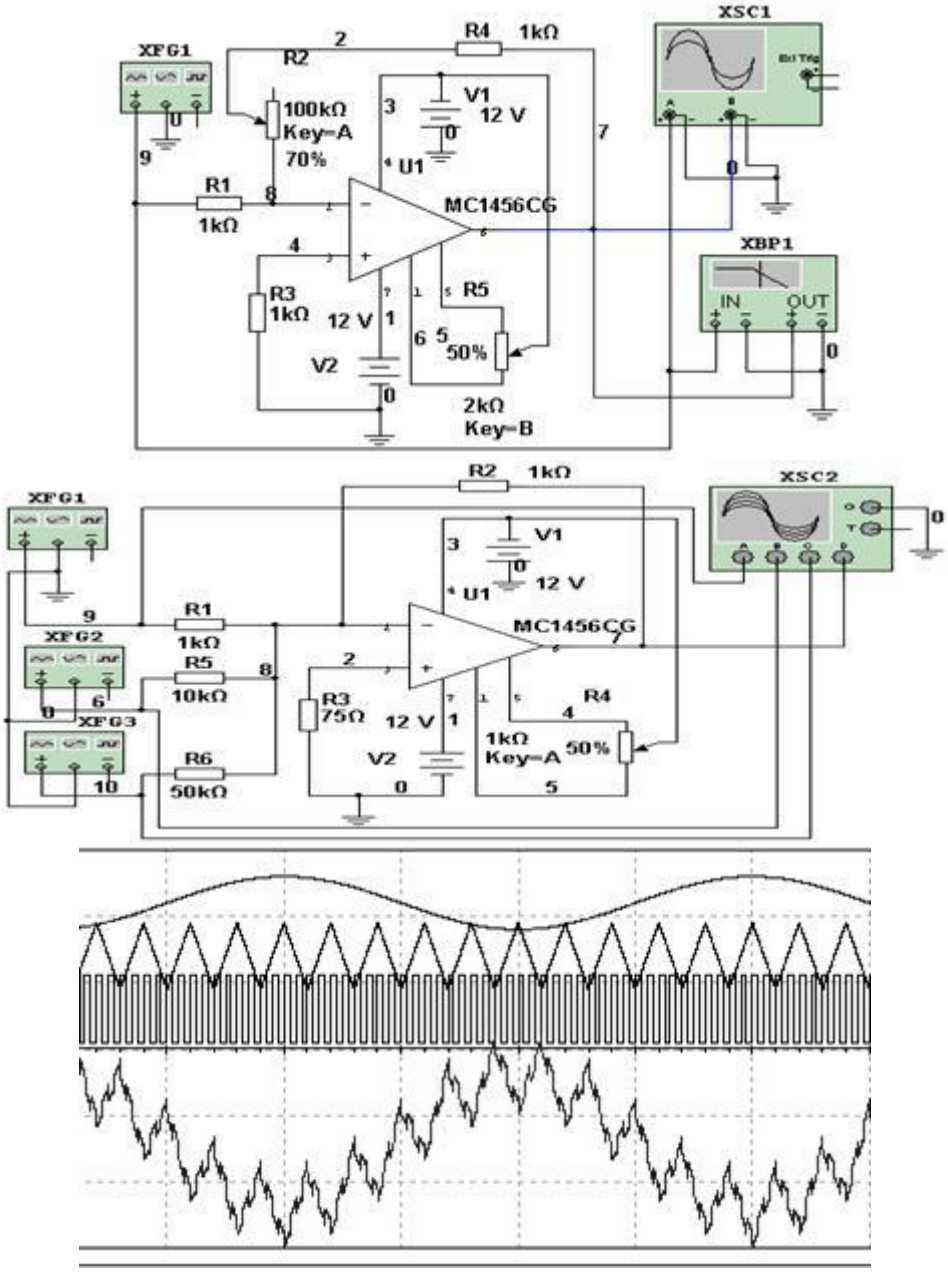
після цього перевищує його за абсолютним значенням, яке відповідає негативному потенціалу на інвертувальному вході ОП ($U_{вх} < 0$), що супроводжується перемиканням компаратора в інший стан, в якому $U^{-}_{вих.max} \approx E^{+}_{C}$. Моменту часу, якщо виконується рівність $U_д = U_{вх.пор}$, відповідає нестійкий режим підсилювача компаратора. При цьому нахил передавальної характеристики визначається власним коефіцієнтом підсилення K_U. Тому відсутність в ОП негативного зворотного зв'язку сприяє підвищенню швидкості перемикання компаратора.

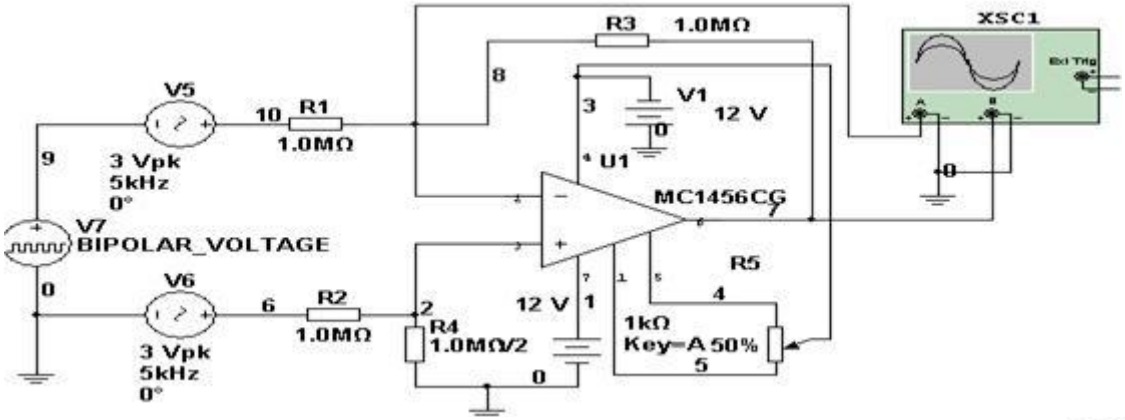
У двовходовому компараторі сигнали, які порівнюють, надходять до обох входів ОП. Тому стан виходу компаратора (полярність вихідної напруги) визначається більшою за рівнем напругою одного з входів, що відображено передавальною характеристикою компаратора. Якщо вхідні напруги однакові, вихідна напруга компаратора дорівнюнулю аналогічно роботі інтегрального ОП. Рівень вхідної напруги компаратора обмежується допустимою синфазною напругою ОП.

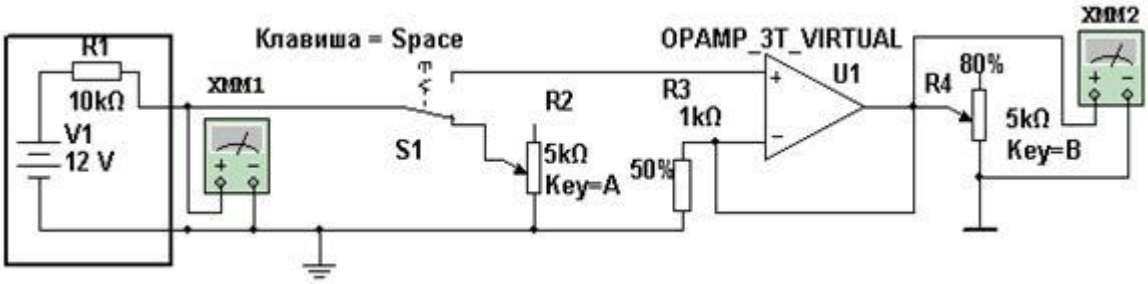


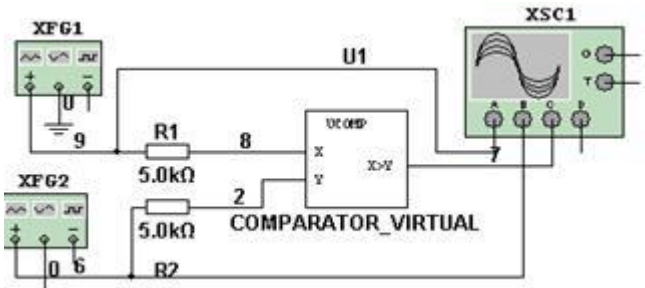


Основним показником ОП, що працюють в імпульсному режимі, є їх швидкодія, яка оцінюється затримкою спрацьовування та часом зростання вихідної напруги. Найбільшу швидкість мають спеціалізовані ОП, що отримали загальну назву “компаратори”, які призначені для імпульсного режиму роботи. Затримка спрацьовування таких мікросхем менше 1 мкс, а час зростання вихідної напруги становить соті частки мікросекунди.

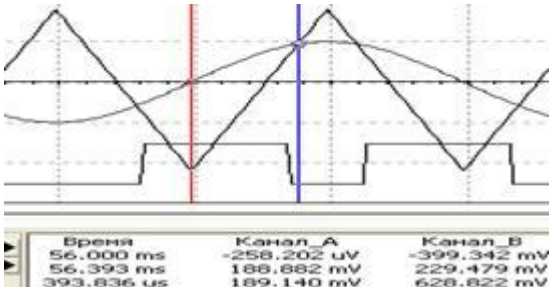








мВ .



4.3. Аналого-цифрові та цифро-аналогові перетворювачі.

Аналого-цифрові перетворювачі (АЦП) це пристрої, які приймають вхідні аналогові сигнали та генерують відповідні до них цифрові сигнали, які придатні для обробки мікропроцесорами та іншими цифровими пристроями.

Принципово не виключена можливість безпосереднього перетворення різних фізичних величин в цифрову форму, однак це завдання вдається розв’язати тільки досить рідко через складність таких перетворювачів. Тому зараз найраціональнішим вважається спосіб перетворення різних за фізичною природою величин спочатку в функціонально пов'язані з ними електричні, а потім уже за допомогою перетворювачів напруга - код – в цифрові. Саме ці перетворювачі і мають на увазі, коли говорять про АЦП.

Процедура аналого-цифрового перетворення неперервних сигналів, яку реалізовують за допомогою АЦП, це перетворення неперервної функції

часу $U(t)$, яка описує вхідний сигнал, у послідовність чисел {$U'(t_j)$}, j =0,1,2,…, що віднесені до деяких фіксованих моментів часу. Цю процедуру можна розділити на дві самостійні операції: дискретизацію і квантування.

Найпоширенішою формою дискретизації, як зазначалось, є рівномірна дискретизація, в основі якої лежить теорема відліків. Згідно з цією теоремою як коефіцієнти a_j потрібно використовувати миттєві значення сигналу $U(t_j)$ в дискретні моменти часу $t_j = j\omega t$, а період дискретизації вибирати з умови:

$$t = 1/2F_m,$$

де F_m – максимальна частота спектра сигналу, що перетворюється.

Тоді отримаємо відомий вираз теореми відліків

$$U(t) = \sum_{j=-\infty}^{\infty} U(j\omega\Delta t)\frac{\sin[2\pi F_m(t - j\Delta t)]}{2\pi F_m(t - j\Delta t)}.$$

Для сигналів зі строго обмеженим спектром цей вираз є тотожністю. Однак спектри реальних сигналів прямують до нуля тільки асимптотично. Застосування рівномірної дискретизації до таких сигналів викликає виникнення в системах обробки інформації специфічних високочастотних спотворень, які зумовлені вибіркою. Для зменшення цих спотворень необхідно або збільшувати частоту дискретизації, або використовувати перед АЦП додатковий фільтр нижніх частот, який обмежуватиме спектр вхідного сигналу перед його аналого-цифровим перетворенням.

У загальному випадку вибір частоти дискретизації буде залежати також від вигляду функції $f_j(t)$, що використовується в першій формулі розділу та допустимого рівня похибок, які виникають при відновленні початкового сигналу за його відліками. Усе це необхідно враховувати при виборі частоти дискретизації, яка визначає необхідну швидкодію АЦП. Часто цей параметр задають розробнику АЦП.

Розглянемо докладніше місце АЦП при виконанні операції дискретизації.

Для достатньо вузькосмугових сигналів операцію дискретизації можна виконувати за допомогою самих АЦП і суміщати таким чином з операцією квантування. Основною закономірністю такої дискретизації є те, що за рахунок скінченного часу одного перетворення та невизначеності моменту його закінчення, який, у загальному випадку, залежить від параметрів вхідного сигналу, не вдається отримати однозначної відповідності між значеннями відліків та моментами часу, до яких їх потрібно віднести. В результаті при роботі із сигналами, які змінюються в часі, виникають специфічні похибки, динамічні за своєю природою, для оцінки яких вводять поняття апертурної невизначеності, яка переважно характеризується апертурним часом.

Апертурним часом t_a називають час, протягом якого зберігається невизначеність між значенням вибірки та часом, до якого вона відноситься. Ефект апертурної невизначеності проявляється або як похибка миттєвого значення сигналу при заданих моментах вимірювання, або як похибка моменту часу, в який проводиться вимірювання при заданому миттєвому значенні сигналу. При рівномірній дискретизації наслідком апертурної невизначеності є виникнення амплітудних похибок, які називаються апертурними та чисельно рівні приростові сигналу протягом апертурного часу.

Якщо використовувати іншу інтерпретацію ефекту апертурної невизначеності, то її наявність викликає "тремтіння" істинних моментів часу, в які беруться відліки сигналу, відносно моментів, які рівновіддалені на осі часу. В результаті замість рівномірної дискретизації зі строго постійним періодом проводиться дискретизація с флюктуючим періодом повторення. Це викликає порушення умов теореми відліків та появи уже розглянутих апертурних похибок в системах цифрової обробки інформації.

Таке значення апертурної похибки можна визначити, розклавши вираз для вхідного сигналу в ряд Тейлора в околі точок відліку, який для j-ї точки має вигляд:

$$U(t) = U(t_j) + t_a U'(t_j) + \frac{t_a^2}{2} U''(t_a) + \ldots$$

та в першому наближенні дає апертурну похибку:

$$\Delta U_a(t_j) \approx t_a U'(t_j),$$

де t_a – апертурний час, який для розглянутого випадку в першому наближенні є часом перетворення АЦП.

Зазвичай для оцінки апертурних похибок використовують синусоїдальний випробувальний сигнал $U(t) = U_m \sin \omega t$.

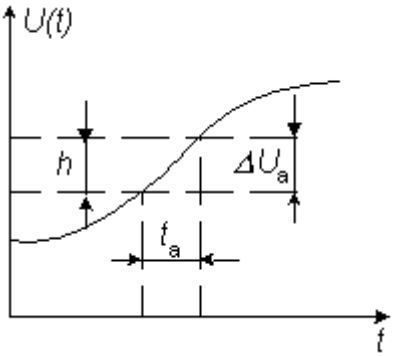


Рисунок 4.4 – Утворення апертурної похибки для випадку, коли вона дорівнює крокові квантування

Якщо прийняти, що для N-розрядного АЦП з роздільною здатністю 2^{-N} апертурна похибка не повинна перевищувати кроку квантування (рис. 4.4),

то залежність між частотою сигналу ω, апертурним часом t_a та відносною апертурною похибкою буде такою: $1/2^N = \omega t_a$.

Для забезпечення дискретизації синусоїдального сигналу частота якого 100 кГц з похибкою 1% час перетворення АЦП повинен бути рівним 25 нс. У той же час за допомогою такого швидкодіючого АЦП принципово можна дискретизувати сигнали, які мають ширину спектра біля 20 МГц. Таким чином, дискретизація за допомогою самого АЦП викликає суттєве розходження вимог між швидкодією АЦП та періодом дискретизації. Це розходження досягає 2...3 порядків та дуже ускладнює і здорожує процес дискретизації, оскільки навіть для порівняно вузькополосних сигналів потребує досить швидкодіючих АЦП. Для достатньо широкого класу сигналів, які швидко змінюються, цю проблему вирішують за допомогою пристроїв вибірки-зберігання, що мають малий апертурний час.

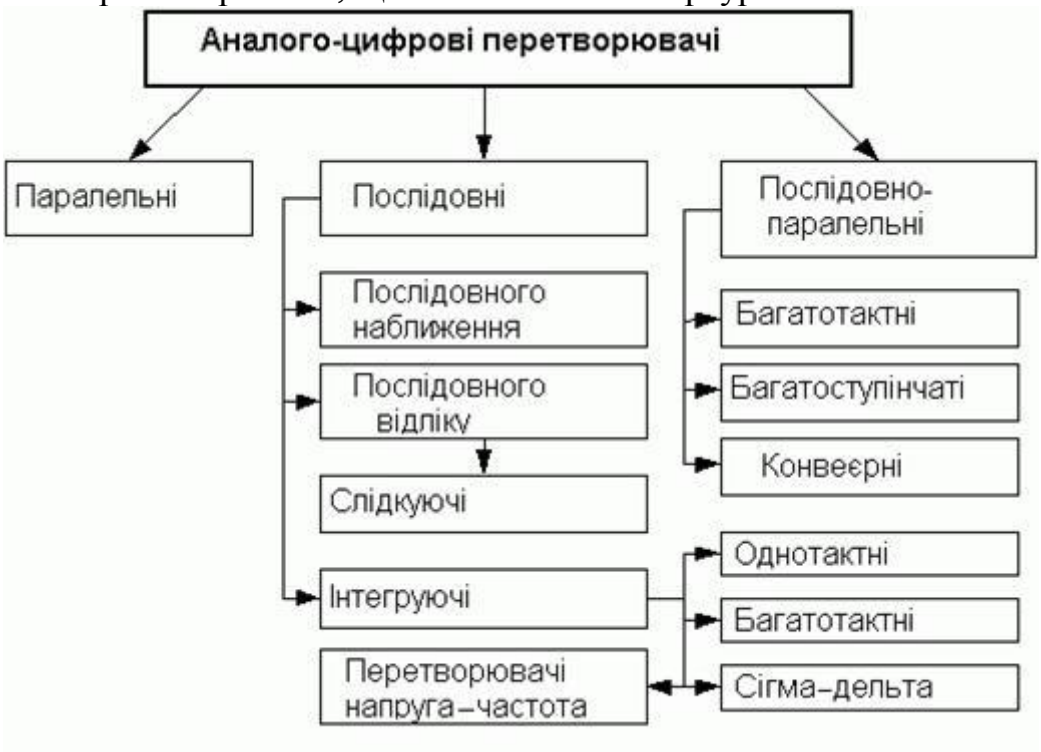


Рисунок 4.5 – Класифікація аналогово-цифрових перетворювачів

Зараз відома велика кількість методів перетворення напруга – код. Ці методи суттєво відрізняються один від одного потенційною точністю, швидкістю перетворення та складністю апаратної реалізації. На рис. 4.5 наведена класифікація АЦП за методами перетворення.

В основу класифікації АЦП покладено ознаку, яка вказує на те, як в часі розгортається процес перетворення аналогової величини в цифрову. В основі перетворення вибіркових значень сигналу в цифрові еквіваленти лежать операції квантування та кодування. Вони можуть проводитись за допомогою або послідовної, або паралельної, або послідовно-паралельної процедур наближення цифрового еквівалента до перетворюваної величини.

Розглянемо детальніше найбільш поширені типи АЦП.

Функціонування аналого-цифрового перетворення за методом послідовного підрахунку можна проілюструвати за допомогою структурної схеми на рис. 4.6.

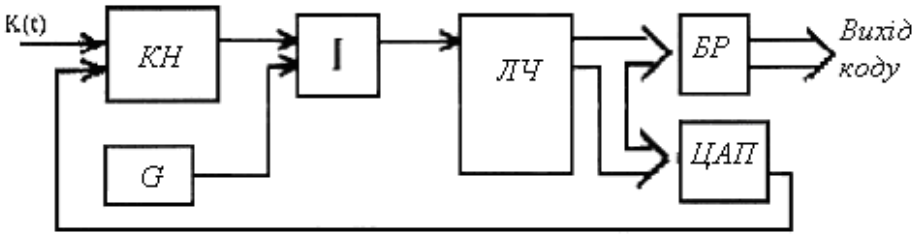


Рисунок 4.6 – АЦП послідовного підрахунку

До складу схеми входять: генератор тактових сигналів (G), компаратор напруги (КН), схема I, лічильник (ЛЧ), буферний регістр (БР), цифро-аналоговий перетворювач (ЦАП). Схема працює наступним чином. На вхід перетворювача подається аналоговий сигнал $x(t)$, який підключається до одного з входів компаратора напруги КН. На другий вхід компаратора подається еталонна напруга (U_{em}), яка формується на виході ЦАП під управлінням колового слова на виході ЛЧ. Компаратор формує на своєму виході сигнал або логічної одиниці, або логічного нуля в залежності від того, яке значення більше. Якщо $U_{em} < x(t)$, то на виході компаратора формується одиниця, яка дозволяє проходження імпульсів з тактового генератора через схему I на лічильний вхід лічильника ЛЧ. На виході лічильника йде процес підрахунку цих імпульсів в двійковому коді від 2^0 до 2^{n-1}. Двійковий код з ЛЧ подається на вхід ЦАП, на виході якого формується ступінчатий сигнал U_{em}. Кожна сходинка цього сигналу відповідає за рівнем інтервалу дискретизації q. Сигнал U_{em} порівнюється із сигналом $x(t)$ і в момент, коли $x(t)$ стає меншим за U_{em}, на виході компаратора формується сигнал логічного нуля. Схема I закривається, лічильник зупиняє підрахунок і набраний двійковий код переписується у вихідний буферний регістр БР для видачі користувачу.

Метод безпосереднього зчитування реалізовується за допомогою так званого АЦП паралельної дії. Такий перетворювач має лінійку 2^{n-1} компараторів напруги, перші входи яких запаралелені і на них подається сигнал $x(t)$. На інші входи під’єднуються виходи подільника еталонної напруги. Виходи компараторів під’єднані до перетворювача одиничного коду в двійковий. Процес перетворення здійснюється за один такт, причому на виході лінійки компараторів до компаратора, який зафіксує $x(t) < U$, буде хвиля одиниць, а далі хвиля нулів одиничного коду. Структурно-функціональна схема перетворення зображена на рис. 4.7, а часова діаграма аналогічна рис. 4.9.

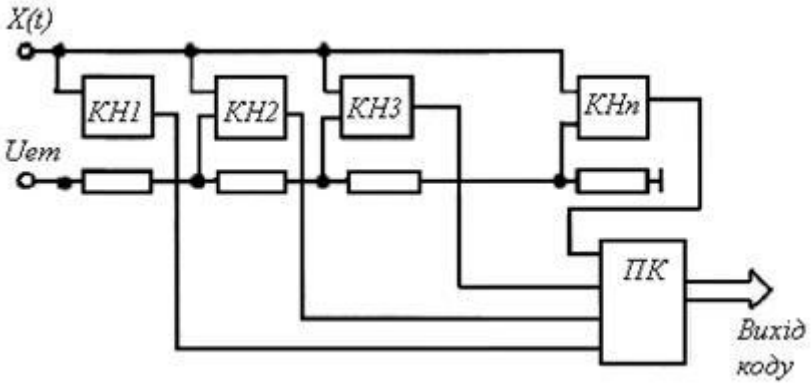


Рисунок 4.7 – АЦП безпосереднього зчитування

Найбільше поширення отримав метод порозрядного зрівноваження, який забезпечує час перетворення від 1 мкс до 1 мс. Структурно-функціональна схема перетворення зображена на рис. 4.8, а часова діаграма – на рис. 4.9.

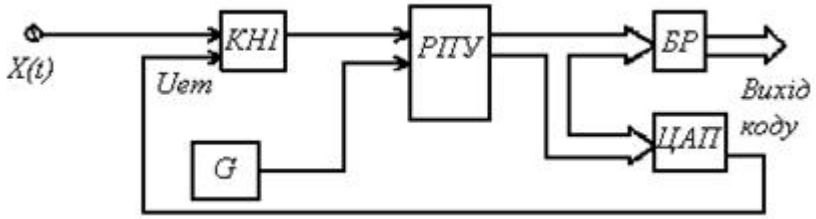


Рисунок 4.8 – АЦП порозрядного зрівноваження

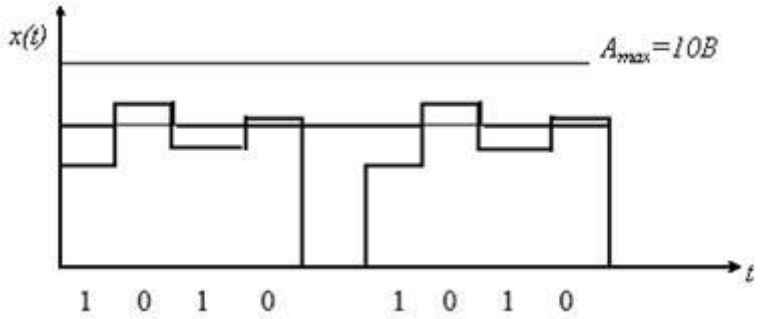


Рисунок 4.9 – Часова діаграма перетворення АЦП за методом порозрядного зрівноваження при $n = 4$

Схема працює так. На вхід АЦП подається вхідний сигнал $x(t)$, який порівнюється з еталонним сигналом U_{em}, що формується на виході ЦАП. ЦАП складається із сукупності $3n$ еталонних джерел сигналів, які управляються за допомогою спеціального регістра порозрядного зрівноваження (РПУ). Перетворення проходить за n часових тактових інтервалів. Причому на першому такті РПУ примусово вмикає в роботу перший розряд ЦАП. Значення першого розряду еталонних величин на виході ЦАП дорівнює половині діапазону перетворення сигналу. Потім в кінці першого тактового інтервалу компаратор проводить порівняння $x(t)$ з U_{em}. Якщо $x(t) < U_{em}$, то примусово увімкнений старший розряд ЦАП залишається ввімкненим до закінчення процесу перетворення. Це забезпечується під управлінням певного сигналу на виході компаратора (1чи

0). Якщо ж $x(t) > U_{em}$, то перший розряд вимикається на початку другого такту. На початку другого такту в роботу примусово вмикається другий розряд ЦАП і знову проводиться порівняння $x(t)$ з U_{em}. Процедура повторюється доти, поки всі розряди ЦАП не візьмуть участі у процесі зрівноваження. В результаті на виході АЦП формується код, що відповідає вхідному сигналу.

Інтегруючі АЦП

Відомо, що недоліком послідовних АЦП є низька завадостійкість результатів перетворення. Дійсно, вибірка миттєвого значення вхідної напруги, переважно включає доданок у вигляді миттєвого значення завади. Згодом при цифровій обробці послідовності вибірок ця складова може бути подавлена, однак на це потрібен час та обчислювальні ресурси. Переважно у АЦП вхідний сигнал інтегрується або неперервно, або у певному часовому діапазоні, тривалість якого зазвичай вибирається кратною періодові завади. Це дозволяє в багатьох випадках приглушити заваду ще на етапі перетворення. Платою за це є понижена швидкодія інтегруючих АЦП.

Спрощена схема АЦП, який працює в два основних такти (АЦП двотактного інтегрування), наведена на рис. 4.10.

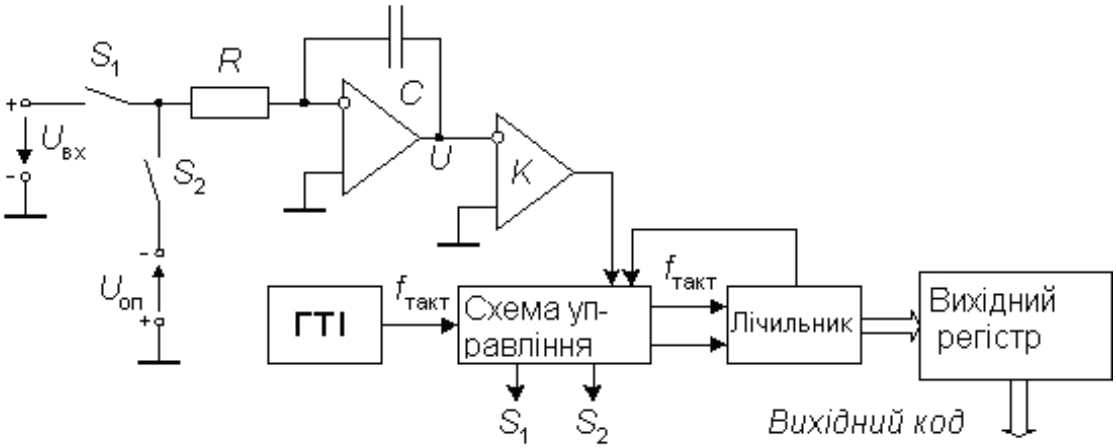


Рисунок 4.10 – Спрощена схема АЦП двотактного інтегрування

Перетворення проходить протягом двох стадій: стадії інтегрування та стадії підрахунку. На початку першої стадії ключ S_1 замкнутий, а ключ S_2 розімкнутий. Інтегратор I інтегрує вхідну напругу $U_{вх}$. Час інтегрування вхідної напруги t_1 постійний; як таймер використовується лічильник з коефіцієнтом підрахунку $K_л$, так, що

$$t_1 = K_л / f_{такт}.$$

До моменту закінчення інтегрування вихідна напруга інтегратора складає

$$U_{вих}(t_1) = -\frac{1}{RC}\int_0^{t_1} U_{вх}(t)dt = -\frac{U_{вх.сер} K_л}{f_{такт} RC},$$

де $U_{вх.сер}$ – середнє за час t_1 значення вхідної напруги.

Після закінчення стадії інтегрування ключ S_1 розмикається, а ключ S_2 замикається та опорна напруга $U_{оп}$ надходить на вхід інтегратора. При цьому вибирається опорна напруга, протилежна за знаком вхідній напрузі. На стадії підрахунку вихідна напруга інтегратора лінійно зменшується за абсолютною величиною, як показано на рис. 4.11.

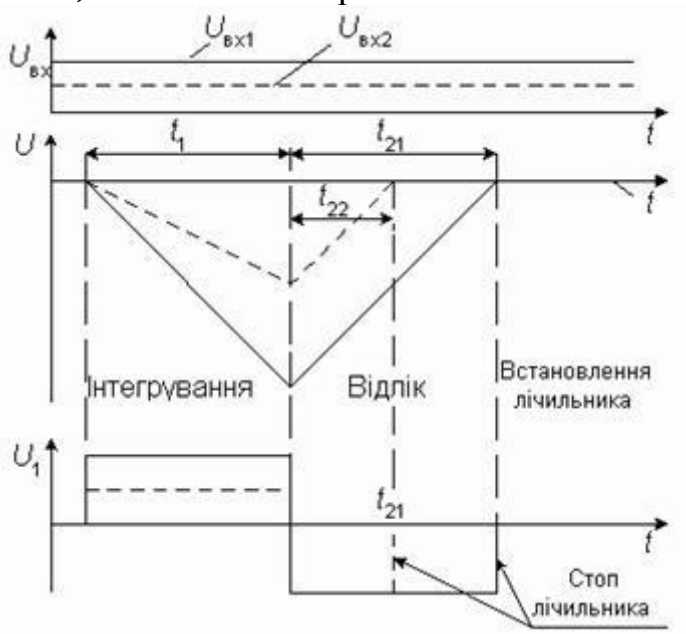


Рисунок 4.11 – Часові діаграми АЦП двотактного інтегрування

Стадія підрахунку закінчується, коли вихідна напруга інтегратора переходить через нуль. При цьому компаратор К переключається та підрахунок зупиняється. Діапазон часу, у якому проходить стадія підрахунку, визначається рівнянням

$$U_{вих}(t_1) + \frac{1}{RC}\int_0^{t_1+t_2} U_{оп}dt = 0.$$

Далі, виконавши прості математичні дії і врахувавши, що:

$$t_2 = \frac{n_2}{f_{такт}},$$

де n_2 – вміст лічильника після закінчення стадії підрахунку,

отримаємо результат

$$n_2 = \frac{U_{вх.сер}K_л}{U_{оп}}.$$

З цієї формули випливає, що відмітною рисою методу багатотактного інтегрування є те, що ні тактова частота, ні постійна інтегрування RC не впливають на результат. Необхідно тільки, щоб тактова частота протягом часу $t_1 + t_2$ залишалася постійною. Це можна забезпечити при використанні простого тактового генератора, оскільки істотні часові чи температурні дрейфи частоти відбуваються за час який більший, ніж час перетворення.

При виведенні попередніх виразів ми бачили, що в остаточний результат входять не миттєві значення перетворюваної напруги, а тільки значення, усереднені за час t_1. Тому змінна напруга послабляється тим сильніше, чим вища її частота.

Визначимо коефіцієнт передачі завади $K_П$ для АЦП двотактного інтегрування. Нехай на вхід інтегратора надходить гармонічний сигнал одиничної амплітуди частотою f з довільною початковою фазою φ. Середнє значення цього сигналу за час інтегрування t_1 дорівнює

$$U_{ñð} = \frac{1}{t_1}\int_0^{t_1} \sin(2\pi ft + \varphi)\,dt = \frac{\sin(\pi ft_1 + \varphi)\sin \pi ft_1}{\pi ft_1}.$$

Коли ця величина досягає максимуму за модулем, то

$$K_П = \left|\frac{\sin^2 \pi ft_1}{\pi ft_1}\right|.$$

Частотна характеристика коефіцієнта приглушення завад АЦП двотактного інтегрування наведена на рис. 4.12.

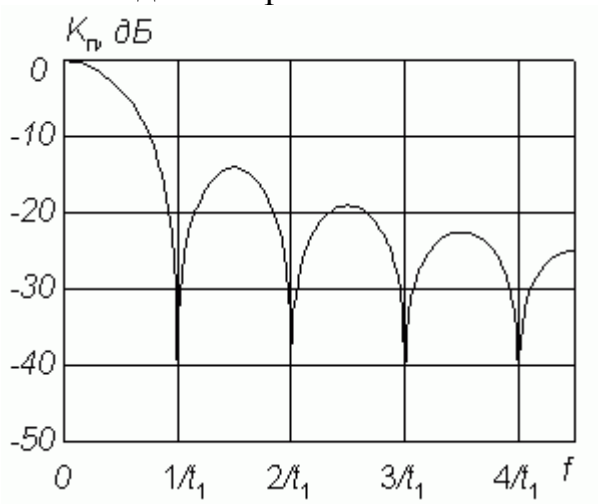


Рисунок 4.12 – Частотна характеристика коефіцієнта приглушення завад АЦП двотактного інтегрування

Як бачимо змінна напруга, період якої в ціле число раз менший t_1, приглушується зовсім Тому доцільно вибрати тактову частоту такою, щоб добуток $K_п \cdot f_{такт}$ був рівним чи кратним періоду напруги промислової мережі.

Багатоканальні АЦП

Багатоканальні АЦП на сьогодні досить поширені, особливо там, де потрібно об’єднати інформацію, отриману від кількох її джерел, тобто, наприклад, від різних сенсорів. Такі АЦП можна застосовувати, наприклад, для моніторингу напруги на входах, контролю крайніх значень, реєстрації показів, управління виходами (навантаженням) тощо. Схема

багатоканального АЦП УМ-АЦП1 на основі мікроконтролера PIC16F876A наведена на рис. 4.13.

Комерційна версія такого пристрою має по 40 входів та виходів, але їх кількість може бути й іншою.

Структурні методи покращення характеристик АЦП

Необхідно відзначити, що висока точність досягається за рахунок як вдосконалення елементної бази, процесу виготовлення, так і застосовуваними матеріалами. Певні відхилення ваг розрядів від необхідних значень зазвичай корегуються шляхом лазерної підгонки в процесі виготовлення резисторів АЦП. При цьому вимагається збільшення площі внутрішньо кристальних компонентів і кристала в цілому, а також виникає проблема вилучення матеріалів кристала в ході пригонки. Ці процеси порушують структуру матеріалів компонентів, зменшують часову і температурну стабільність схеми.

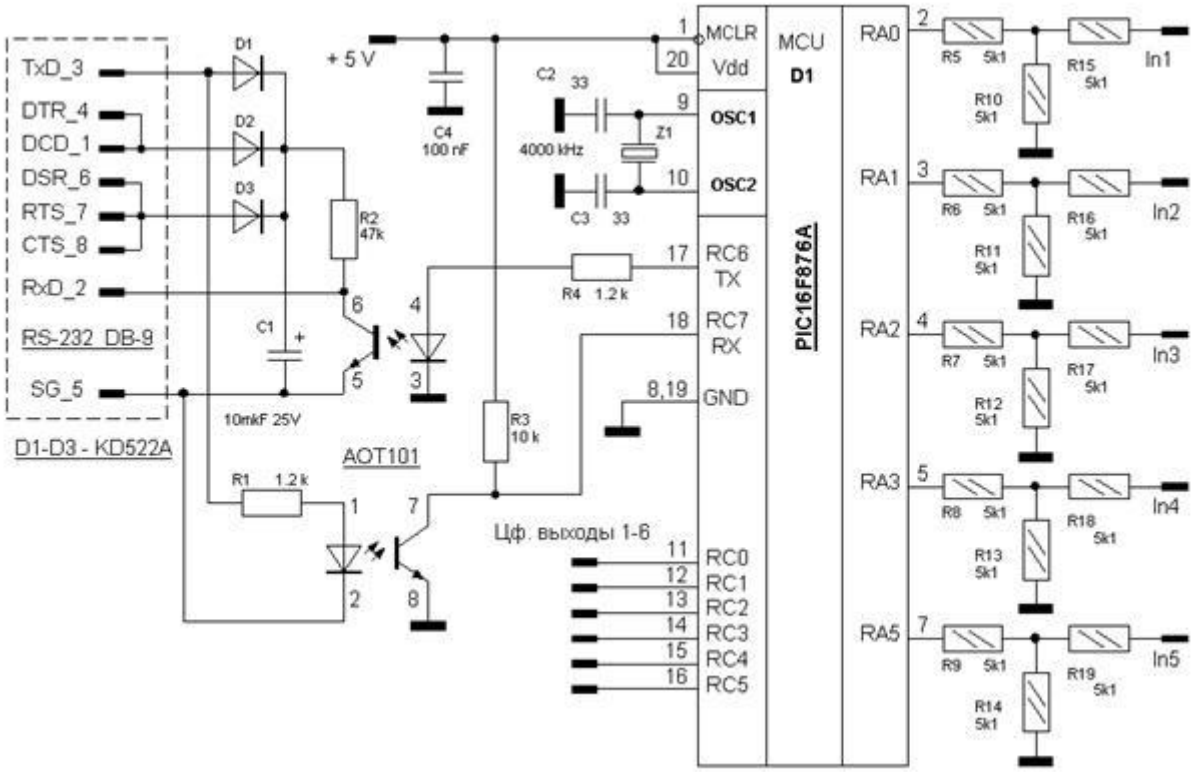


Рисунок 4.13 – Схема багатоканального АЦП УМ-АЦП1

Більш перспективним є підхід, який уникає фізичного впливу на елементи схеми. Наприклад, у випадку використання ЦАП на основі двійкової системи числення, зменшення статичних похибок досягається корекцією вихідної величини шляхом введення поправки в аналоговій формі, що формується додатковим корегувальним ЦАП. В цьому випадку перетворюваний код $K_{вх}$ подається, як показано на рис. 4.14, на вхід основного ЦАП і в цифровий обчислювальний пристрій (ЦОП).

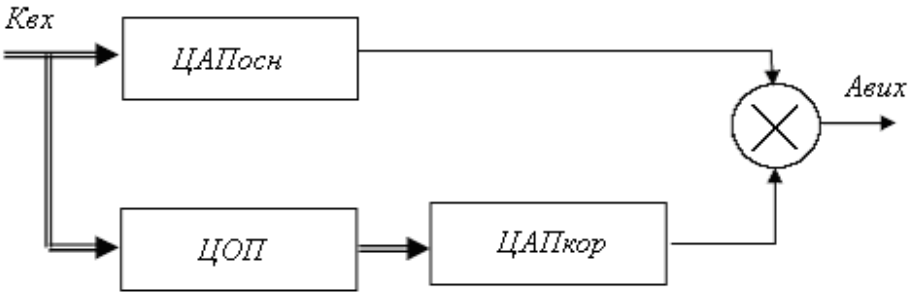


Рисунок 4.14 – Корекція вихідної величини двійкового ЦАП

В ЦОП вираховується код поправки, що надходить на вхід корегувального ЦАП. Результат перетворення $A_{вих}$ формується за допомогою суматора $\otimes$ аналогових величин. Але застосування такого принципу при аналого-цифровому перетворенні дає деяке зниження швидкості пристроїв.

Також використовують введення в прилади при розробці інформаційної надмірності у вигляді надлишкових позиційних систем числення (НПСЧ), що комплексно вдосконалює водночас кілька характеристик аналого-цифрового перетворення. Збільшення розрядної сітки пристрою (а відповідно і збільшення кількості тактів при порозрядному зрівноваженні) підвищує точність АЦП середньої і високої швидкості, реалізованих на грубих аналогових вузлах, а з іншого боку підвищує швидкодію високоточних АЦП на елементній базі середньої швидкості.

Підвищення швидкодії багаторозрядних АЦП досягається двома шляхами. Перший орієнтується на використання більш досконалої елементної бази, що не є досить простим шляхом. Другий шлях пов’язаний із введенням надмірності, переважно структурної, при проектуванні.

Цифро-аналогові перетворювачі

Необхідність здійснення операції відновлення вихідного сигналу з дискретних відліків, а також необхідність здійснення операцій формування еталонних сигналів при аналого-цифровому перетворенні висуває задачу цифро-аналогового перетворення. Суть операції цифро-аналогового перетворення полягає у формуванні аналогових сигналів, що відповідають кодовим словам дискретного сигналу. Технічно це формування здійснюється цифро-аналоговим перетворювачем (ЦАП).

Аналоговий сигнал на виході ЦАП може бути сформований шляхом множення опорної напруги $E_{оп} = q$ на вагові розрядні коефіцієнти кодового слова $a_i = 2^i$, таким чином, що $U_{вих} = q\left(a_0\, 2^0 + a_1\, 2^1 + \ldots + a_{n-1}\, 2^{n-1}\right)$

Технічно найпростіше ЦАП реалізовується на принципі підсумовування розрядних струмів $U_{вих} = \sum I_1 R_{оп} = R_{оп}\left(a_0 I_1 + a_1 I_2 + \ldots + a_n I_n\right)$ (рис. 4.15).

Схема реалізації ЦАП для підсумовування струму містить джерело стабільної напруги E_0, матрицю двійково-зважених резисторів $\left(R \cdot 2^i\right)$, набір

ключів KE_i, що реалізовують розрядні коефіцієнти a_i і перетворювач струму в напругу на операційному підсилювачі ОП.

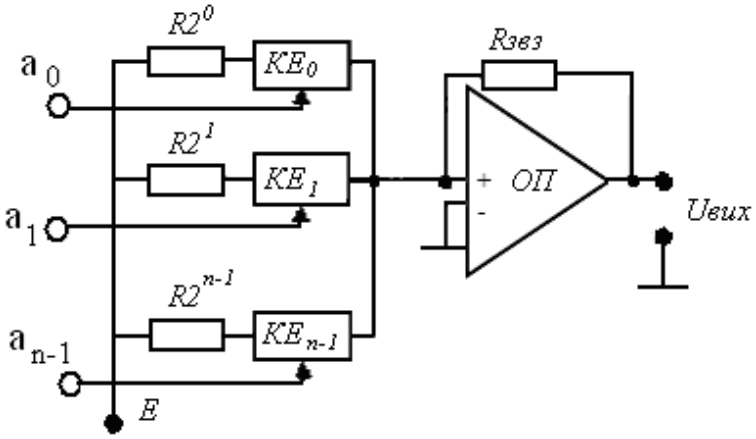


Рисунок 4.15 – ЦАП для підсумовування струму

Часова діаграма класичного процесу цифро-аналогового перетворення має вигляд (рис. 4.16).

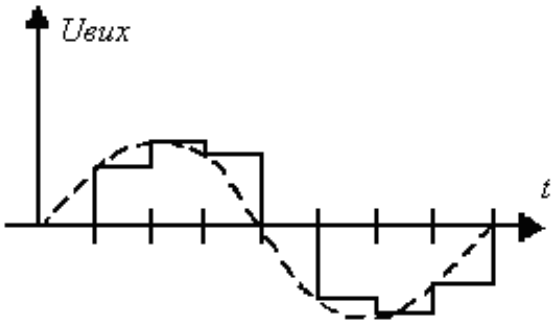


Рисунок 4.16 – Часова діаграма процесу ЦА перетворення

При малій кількості дискретних вибірок миттєвих значень сигналу, цей сигнал мало нагадує вихідний, однак може бути наближеним до нього шляхом аналогової фільтрації або інтерполяції.

Основні типи електронних ЦАП

1. Широтно-імпульсний модулятор – найпростіший тип ЦАП. Стабільне джерело струму чи напругиперіодично вмикається на час, пропорційний перетворюваному цифровому коду, далі отримана імпульсна послідовність фільтрується аналоговим фільтром низьких частот . Такий спосіб часто використовується для керування швидкістю електромоторів, а також стає популярним в Hi-Fi аудіотехніці.

2. ЦАП передискретизації, такі, як дельта-сигма ЦАП, основані на змінюваній густоті імпульсів. Передискретизація дозволяє використовувати ЦАП з меншою розрядністю для досягнення більшої розрядності кінцевого перетворення; часто дельта-сигма ЦАП будується на основі найпростішого однобітового ЦАП, який є практично лінійним. На ЦАП малої розрядності надходить імпульсний сигнал з модульованою густотою імпульсів (з постійною тривалістю імпульсу, але зі змінною шпаруватістю), створений з використанням негативного зворотного зв’язку. Негативний зворотний

зв'язок виступає в ролі фільтра високих частот для шуму квантування. Більшість ЦАП більшої розрядності (більше 16 біт) побудовані на цьому принципі внаслідок його високої лінійності і низької вартості. Швидкодія дельта-сигма ЦАП сягає сотень тисяч відліків в секунду, розрядність – до 24 біт. Для генерації сигналу з модульованою густотою імпульсів можна використати простий дельта-сигма модулятор першого порядку чи більш високого порядку як MASH (англ. Multi stage noise SHaping). Зі збільшенням частоти передискретизації знижуються вимоги до вихідного фільтра низьких частот і поліпшується приглушення шуму квантування.

3. ЦАП зважування, в якому кожному біту перетворюваного двійкового коду відповідає резистор чи джерело струму, підключене до спільної точки додавання. Сила струму джерела (провідність резистора) пропорційна вазі біта, якому він відповідає. Таким чином, всі ненульові біти коду додаються з вагою. Метод зважування – один з найшвидших, але йому властива низька точність через необхідність наявності набору множини різних прецизійних джерел чи резисторів. Через цю причину ЦАП зважування мають розрядність не більше восьми біт.

4. Ланцюгова R-2R схема є варіацією ЦАП зважування. В R-2R ЦАП зважені значення створюються в спеціальній схемі, яка складається з резисторів опорами R і 2R. Це дозволяє суттєво збільшити точність порівняно зі звичайним ЦАП зважування, оскільки порівняно просто виготовити набір прецизійних елементів з однаковими параметрами. Недоліком методу є більш низька швидкість внаслідок паразитної ємності.

5. Сегментний ЦАП містить по одному джерелу струму чи резистору на кожне можливе значення вихідного сигналу. Так, наприклад, восьмибітовий ЦАП цього типу містить 255 сегментів, а 16-бітовий – 65535. Теоретично, сегментні ЦАП мають найбільшу швидкодію, оскільки для перетворення достатньо замкнути один ключ, який відповідає вхідному коду.

6. Гібридні ЦАП використовують комбінацію перерахованих вище способів. Більшість мікросхем ЦАП належать до цього типу, вибір конкретного набору способів є компромісом між швидкодією, точністю і вартістю ЦАП.

Цифрові вимірювачі неелектричних величин

Сприйняття інформації про об'єкти чи процеси здійснюється за допомогою пристроїв, які називаються первинними перетворювачами чи сенсорами. В більшості випадків сенсори відображають вхідну інформацію у вигляді еквівалентного електричного параметра. Тобто, сенсором називається елемент, який приймає контрольований параметр і перетворює його до вигляду, зручного для подальшої обробки (вимірювання, передачі, контролю).

Відповідно до схеми вмикання сенсорів можна визначити дві групи узгоджувально-нормувальних пристроїв. До першої групи належать пристрої, в яких сенсори є елементами подільників напруги (рис. 4.17, а), до

другої – пристрої, в яких сенсори є елементами коливальних систем ВЧ генераторів (рис. 4.17, б).

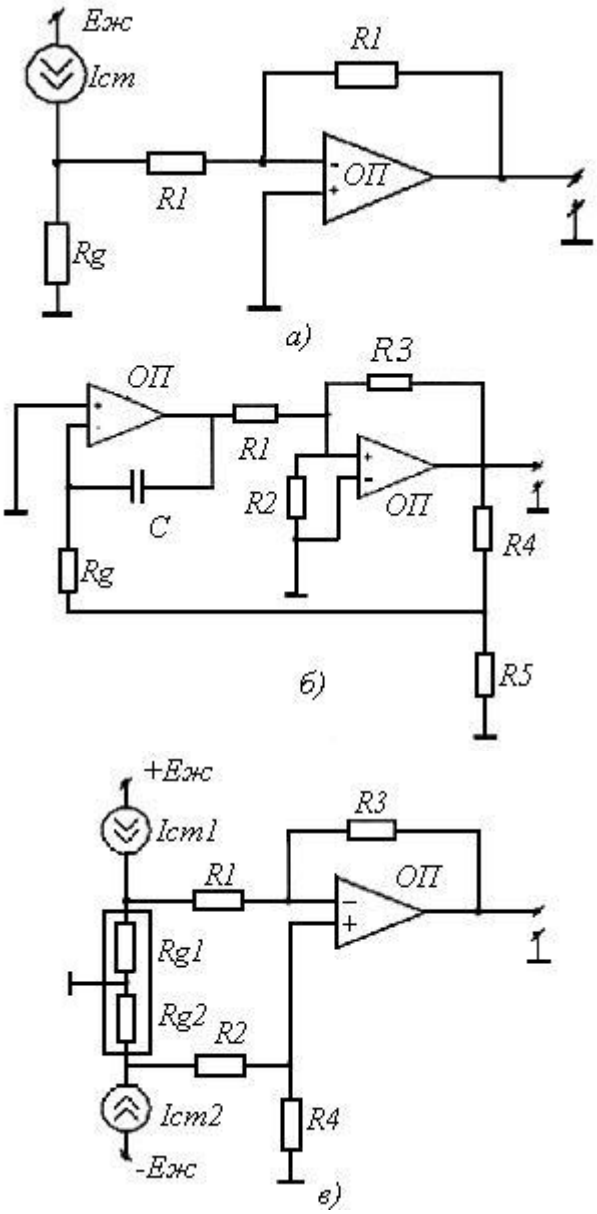
В пристроях першої групи сенсори найчастіше вмикаються за диференціальною чи мостовою схемою (рис. 4.17, в, г, відповідно).

а) з подільниками напруги; б) з елементами ВЧ генераторів; в) за диференціальною схемою увімкнення; г) за мостовою схемою увімкнення

Диференціальні схеми відрізняються високою стабільністю, оскільки дестабілізуючі фактори одночасно діють на обидва елементи диференціального сенсора, що компенсує цей вплив.

Сенсори в мостовій схемі входять до складу моста, який врівноважений при деякому (звичайно нульовому чи початковому) значенні контрольованого параметра.

При вимірюваннях деяких неелектричних величин не завжди вдається перетворити їх безпосередньо в електричну величину. В цих випадках здійснюють подвійне перетворення первинної вимірюваної величини в проміжну неелектричну величину, яку перетворюють потім у вихідну електричну величину.



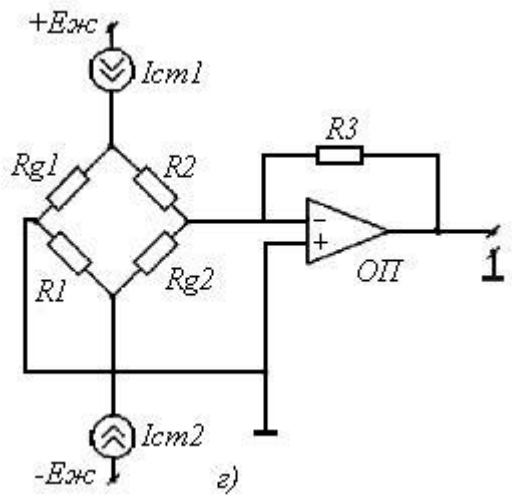


Рисунок 4.17 – Схеми узгодження

Сукупність двох відповідних вимірювальних перетворювачів утворюють комбінований сенсор (рис. 4.18).

Рисунок 4.18 – Блок-схема комбінованого сенсора

Подібні перетворювачі зручні для вимірювання неелектричних (механічних) величин, які викликають в первинному перетворювачі деформацію або переміщення вихідного елемента, до яких чутливий вторинний перетворювач.

ТЕМА 5. СТРУКТУРА І ФУНКЦІЮВАННЯ МІКРОПРОЦЕСОРНИХ СИСТЕМ ДЛЯ ЕЛЕКТРОПОБУТОВОЇ ТЕХНІКИ

Мікропроцесорна система може розглядатися як окремий випадок електронної системи, що призначена для обробки вхідних сигналів і видачі вихідних сигналів (Рис. 1.1). У якості вхідних і вихідних сигналів при цьому можуть використовуватися аналогові сигнали, одиночні цифрові сигнали, цифрові коди, послідовності цифрових кодів. Усередині системи може відбуватися збереження, накопичення сигналів (чи інформації), але суть від цього не змінюється. Якщо система цифрова (а мікропроцесорні системи відносяться до розряду цифрових), то вхідні аналогові сигнали перетворяться в послідовності кодів вибірок за допомогою АЦП, а вихідні аналогові сигнали формуються з послідовності кодів вибірок за допомогою ЦАП. Обробка і збереження інформації відбувається в цифровому виді.

Характерною рисою традиційної цифрової системи є те, що алгоритми обробки і збереження інформації в ній жорстко зв'язані зі схемотехнікою системи. Тобто зміна цих алгоритмів можлива тільки шляхом зміни структури системи, заміни електронних вузлів, які входять до системи, і/або зв'язків між ними. Наприклад, якщо нам потрібна додаткова операція додавання, то необхідно додати в структуру системи зайвий суматор. Або якщо потрібна додаткова функція збереження коду протягом одного такту, то ми повинні додати в структуру ще один регістр. Природно, що це практично неможливо зробити в процесі експлуатації, обов'язково потрібний новий виробничий цикл проектування, виготовлення, налагодження всієї системи. Саме тому традиційна цифрова система часто називається системою на "жорсткій логіці".

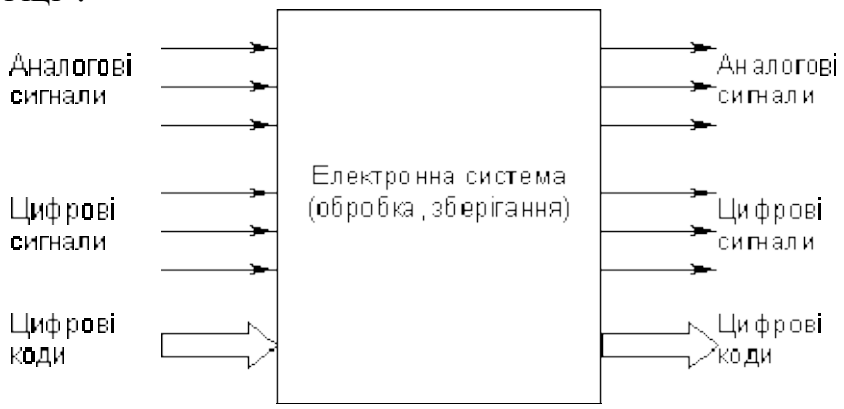


Рис. 5.1. Електронна система.

Будь-яка система на "жорсткій логіці" обов'язково є спеціалізованою систему, яка настроєну винятково на одну задачу чи (рідше) на декілька близьких, заздалегідь відомих задач. Це має свої безперечні переваги:

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

По-друге, саме спеціалізована система може забезпечити максимально високу швидкодію, тому що швидкість виконання алгоритмів обробки

інформації визначається в ній тільки швидкодією окремих логічних елементів і обраною схемою шляхів проходження інформації. А саме логічні елементи завжди володіють максимальною на даний момент швидкодією.

Але в той же час великим недоліком цифрової системи на "жорсткій логіці" є те, що для кожної нової задачі її треба проектувати і виготовляти заново. Це процес тривалий, дорогий і потребує високої кваліфікації виконавців. А якщо розв'язувана задача раптом змінюється, то вся апаратура повинна бути цілком замінена. У нашому швидко мінливому світі це досить марнотратно.

Шлях подолання цього недоліку досить очевидний: треба побудувати таку систему, що могла б легко адаптуватися під будь-яку задачу, перебудовуватися з одного алгоритму роботи на іншій без зміни апаратури. І задавати той чи інший алгоритм ми тоді могли б шляхом введення в систему деякої додаткової керуючої інформації, *програми* роботи системи (Рис. 1.2). Тоді система стане універсальною, або *програмованою*, не жорсткою, а гнучкою. Саме це і забезпечує мікропроцесорна система.

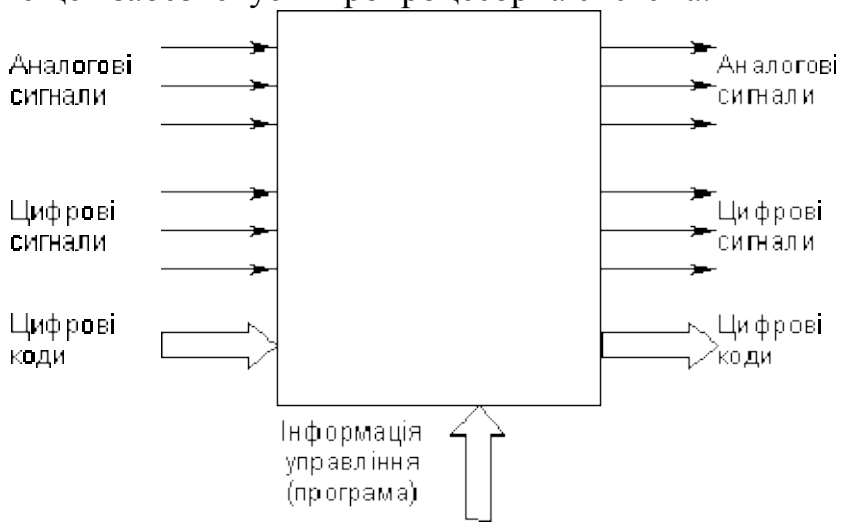


Рис. 5.2. Програмована (вона ж універсальна) електронна система.

Але будь-яка універсальність обов'язково приводить до надмірності. Адже вирішення максимально важкої задачі вимагає набагато більше засобів, ніж рішення максимально простої задачі. Тому складність універсальної системи повинна бути такою, щоб забезпечувати рішення самої важкої задачі, а при рішенні простої задачі система буде працювати далеко не в повну силу, буде використовувати не усі свої ресурси. І чим простішою є розв'язувана задача, тим більша надмірність, і тим менш виправданою стає універсальність. Надмірність веде до збільшення вартості системи, зниження її надійності, збільшення споживаної потужності і т.д.

Крім того, універсальність, як правило, викликає істотне зниження швидкодії. Оптимізувати універсальну систему так, щоб кожна нова задача розв'язувалась максимально швидко, просто неможливо. Загальне правило таке: чим більшою є універсальність, гнучкість, тим менша швидкодія. Більше того, для універсальних систем не існує таких задач (нехай навіть і

найпростіших), які б вони вирішували з максимально можливою швидкодією. За все приходиться платити.

Таким чином, можна зробити наступний висновок. Системи на "жорсткій логіці" доцільна там, де розв'язувана задача не змінюється тривалий час, де потрібно забезпечити найвищу швидкодію, де алгоритми обробки інформації гранично прості. А універсальні, програмовані системи доцільні там, де часто змінюються розв'язувані задачі, де висока швидкодія не надто важлива, де алгоритми обробки інформації складні. Тобто будь-яка система повинна бути на своєму місці.

Однак за останні десятиліття швидкодія універсальних (мікропроцесорних) систем сильно виросла (на кілька порядків). До того ж великий обсяг випуску мікросхем для цих систем викликав різке зниження їх вартості. У результаті області застосування систем на "жорсткій логіці" різко звузилися. Більше того, високими темпами розвиваються зараз програмовані системи, які призначені для рішення однієї задачі або декількох близьких задач. Вони вдало поєднують у собі переваги як систем на "жорсткій логіці", так і програмованих систем, забезпечуючи поєднання досить високої швидкодії і необхідної гнучкості. Так що витіснення "жорсткої логіки" продовжується.

5.1. Що таке мікропроцесор?

Ядром будь-якої мікропроцесорної системи є мікропроцесор або просто процесор (від англійського processor). Перекласти на українську мову це слово вірніше всього як "оброблювач", тому що саме **мікропроцесор** - це той вузол, блок, який робить всю обробку інформації усередині мікропроцесорної системи. Інші вузли виконують усього лише допоміжні функції: збереження інформації (у тому числі і керуючої інформації, тобто програми), зв'язку з зовнішніми пристроями, зв'язку з користувачем і т.д. Процесор замінює практично всю "жорстку логіку", що знадобилася б у випадку традиційної цифрової системи. Він виконує арифметичні функції (додавання, множення і т.д.), логічні функції (зсуву, порівняння, маскування кодів та ін.), тимчасове збереження кодів (у внутрішніх регістрах), пересилання кодів між вузлами мікропроцесорної системи і багато чого іншого. Кількість таких елементарних операцій, що виконуються процесором, може досягати декількох сотень. Процесор можна порівняти з мозком системи.

Але при цьому треба враховувати, що усі свої операції процесор виконує *послідовно*, тобто одну за іншою, по черзі. Звичайно, існують процесори з паралельним виконанням деяких операцій, зустрічаються також мікропроцесорні системи, у яких кілька процесорів працюють над однією задачею паралельно, але це рідкісні винятки. З одного боку, послідовне виконання операцій - безсумнівна перевага, тому що дозволяє за допомогою тільки одного процесора виконувати будь-які, самі складні алгоритми обробки інформації. Але, з іншого боку, послідовне виконання операцій приводить до того, що час виконання алгоритму залежить від його

складності. Прості алгоритми виконуються швидше складних. Тобто мікропроцесорна система здатна зробити все, але працює вона не занадто швидко, адже всі інформаційні потоки приходиться пропускати через один-єдиний вузол - мікропроцесор (Рис. 1.3). У традиційній цифровій системі можна легко організувати паралельну обробку всіх потоків інформації, щоправда, ціною ускладнення схеми.

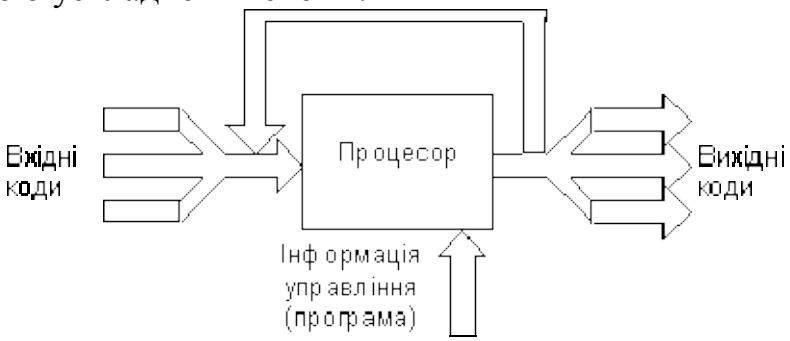


Рис. 5.3. Інформаційні потоки в мікропроцесорній системі.

Отже, мікропроцесор здатний виконувати безліч операцій. Але відкіля він довідається, яку операцію йому треба виконувати в даний момент? Саме це визначається керуючою інформацією - програмою. Програма - це набір *команд (інструкцій)*, тобто цифрових кодів, розшифрувавши які, процесор довідається, що йому треба робити. Програма від початку і до кінця складається людиною, програмістом, а процесор виступає в ролі слухняного виконавця цієї програми, ніякої ініціативи він не виявляє (якщо, звичайно, справний). Тому порівняння процесора з мозком не занадто коректно. Він усього лише виконавець того алгоритму, що заздалегідь склала для нього людина. Будь-яке відхилення від цього алгоритму може бути викликано тільки несправністю процесора або яких-небудь інших вузлів мікропроцесорної системи.

Усі команди, які можуть виконуватись процесором, утворюють *систему команд* процесора. Структура й обсяг системи команд процесора визначають його швидкодію, гнучкість, зручність використання. Усього команд у процесора може бути від декількох десятків до декількох сотень. Система команд може бути розрахована на вузьке коло розв'язуваних задач (у спеціалізованих процесорів) або на максимально широке коло задач (в універсальних процесорів). Коди команд можуть мати різну кількість розрядів (займати від одного до декількох байт). Кожна команда має свій час виконання, тому час виконання всієї програми залежить не тільки від кількості команд у програмі, але і від того, які саме команди використовуються.

Для виконання команд у структуру процесора входять внутрішні регістри, арифметико-логічний пристрій (АЛП, ALU - Arithmetic Logic Unit), мультиплексори, буфери, регістри й інші вузли. Робота усіх вузлів синхронізується загальним зовнішнім тактовим сигналом процесора. Тобто процесор є досить складним цифровим пристроєм (Рис. 1.4).

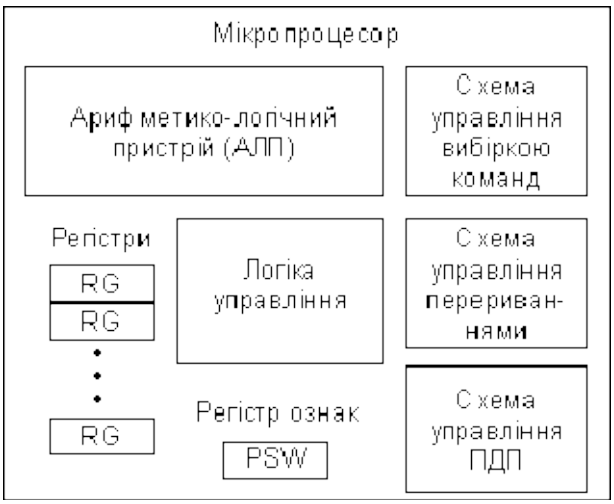


Рис. 5.4. Приклад структури найпростішого процесора.

Утім, для розробника мікропроцесорних систем інформація про тонкості внутрішньої структури процесора не надто важлива. Розробник повинен розглядати процесор як "чорний ящик", який у відповідь на вхідні і керуючі коди робить ту або іншу операцію і видає вихідні сигнали. Розробникові необхідно знати систему команд, режими роботи процесора, а також правила взаємодії процесора з зовнішнім світом або, як їх ще називають, *протоколи обміну інформацією*. Про внутрішню структуру процесора треба знати тільки те, що необхідно для вибору тієї чи іншої команди, того чи іншого режиму роботи.

5.2. Шинна структура зв'язків

Для досягнення максимальної універсальності і спрощення протоколів обміну інформацією в мікропроцесорних системах застосовується так звана шинна структура зв'язків між окремими пристроями, що входять у систему. Суть шинної структури зв'язків зводиться до наступного:

При класичній структурі зв'язків (Рис. 1.5) усі сигнали і коди між пристроями передаються окремими лініями зв'язку. Кожен пристрій, який входить у систему, передає свої сигнали і коди незалежно від інших пристроїв. При цьому в системі виходить дуже багато ліній зв'язку і різних протоколів обміну інформацією.

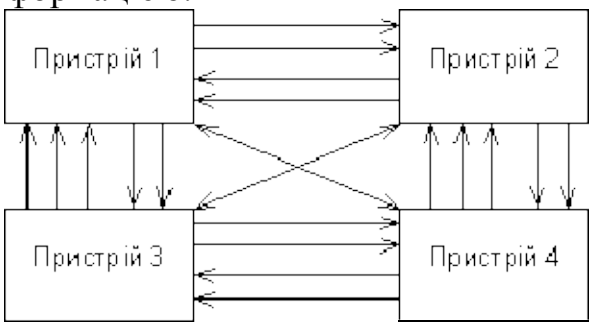


Рис. 5.5. Класична структура зв'язків.

При шинній структурі зв'язків (Рис. 1.6) усі сигнали між пристроями передаються одними і тими ж лініями зв'язку, але в різний час (це

називається мультиплексованою передачею). Причому передача всіма лініями зв'язку може здійснюватися в обидва напрямки (так звана двонапрямлена передача). У результаті кількість ліній зв'язку істотно скорочується, а правила обміну (протоколи) спрощуються. Група ліній зв'язку, якими передаються сигнали або коди саме і називається **шиною** (англ. bus).

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

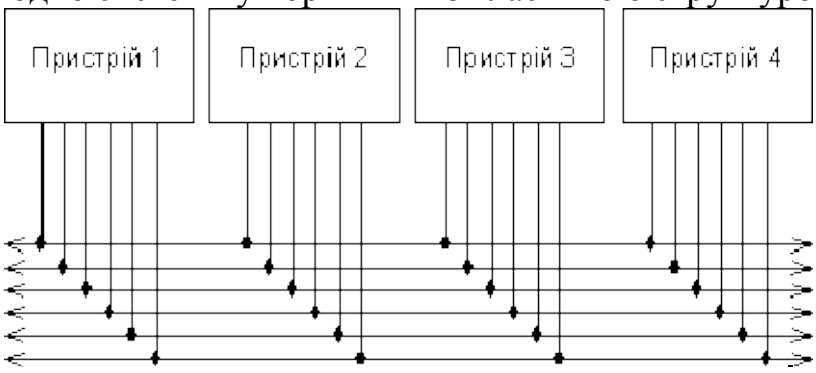


Рис. 5.6. Шинна структура зв'язків.

Велика перевага шинної структури зв'язків полягає в тому, що всі пристрої, які під'єднані до шини, повинні приймати і передавати інформацію за тими самими правилами (протоколам обміну інформацією із шини). Відповідно, усі вузли, які відповідають за обмін із шиною в цих пристроях, повинні бути однакові, уніфіковані.

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

У системах із шинною структурою зв'язків застосовують усі три існуючі різновиди вихідних каскадів цифрових мікросхем:

- стандартний вихід або вихід із двома станами (позначається 2С, 2S, рідше ТТЛ, TTL);
- вихід з відкритим колектором (позначається ВК, ОК, ОС);
- вихід із трьома станами або (що те ж саме) з можливістю відключення (позначається 3С, 3S).

Спрощено ці три типи вихідних каскадів можуть бути подані у виді схем на Рис. 1.7.

У виходу 2С два ключі замикаються по черзі, що відповідає рівням логічної одиниці (верхній ключ замкнуто) і логічного нуля (нижній ключ замкнуто). У виходу ВК замкнутий ключ формує рівень логічного нуля,

розімкнутий - логічної одиниці. У виходу ЗС ключі можуть замикатися по черзі (як у випадку 2С), а можуть розмикатися одночасно, утворюючи тим самим третій, високоімпедансний, стан. Перехід у третій стан (Z-стан) керується сигналом на спеціальному вході EZ.

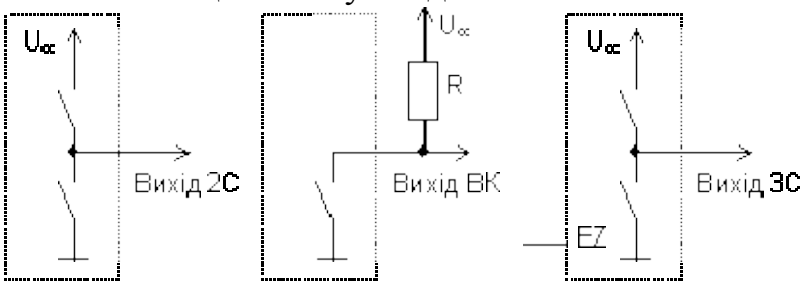


Рис. 5.7. Три типи виходів цифрових мікросхем.

Вихідні каскади типів ЗС та ВК дозволяють з'єднувати кілька виходів мікросхем для отримання мультиплексованих (Рис. 1.8) або двонапрямлених (Рис. 1.9) ліній.

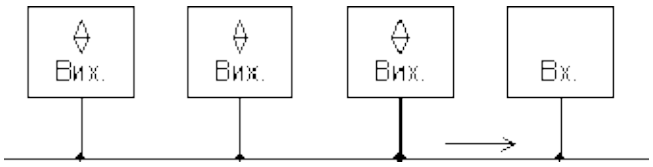


Рис. 5.8. Мультиплексована лінія.

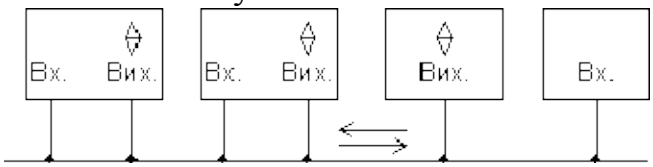


Рис. 5.9. Двонапрямлена лінія.

При цьому у випадку виходів ЗС необхідно забезпечити, щоб на лінії завжди працював тільки один активний вихід, а всі інші виходи знаходилися б у цей час у третьому стані, інакше можливі конфлікти. Об'єднані виходи ВК можуть працювати всі одночасно, без усяких конфліктів.

Типова структура мікропроцесорної системи наведена на Рис. 1.10. Вона містить у собі три основних типи пристроїв:

- **процесор**;
- **пам'ять**, що включає оперативну пам'ять (ОЗП, RAM - Random Access Memory) і постійну пам'ять (ПЗП, ROM -Read Only Memory), що служить для збереження даних і програм;
- **пристрою вводу/виводу** (ПВВ, I/O - Input/Output Devices), які служать для зв'язку мікропроцесорної системи з зовнішніми пристроями, для прийому (уведення, читання, Read) вхідних сигналів і видачі (виведення, запису, Write) вихідних сигналів.

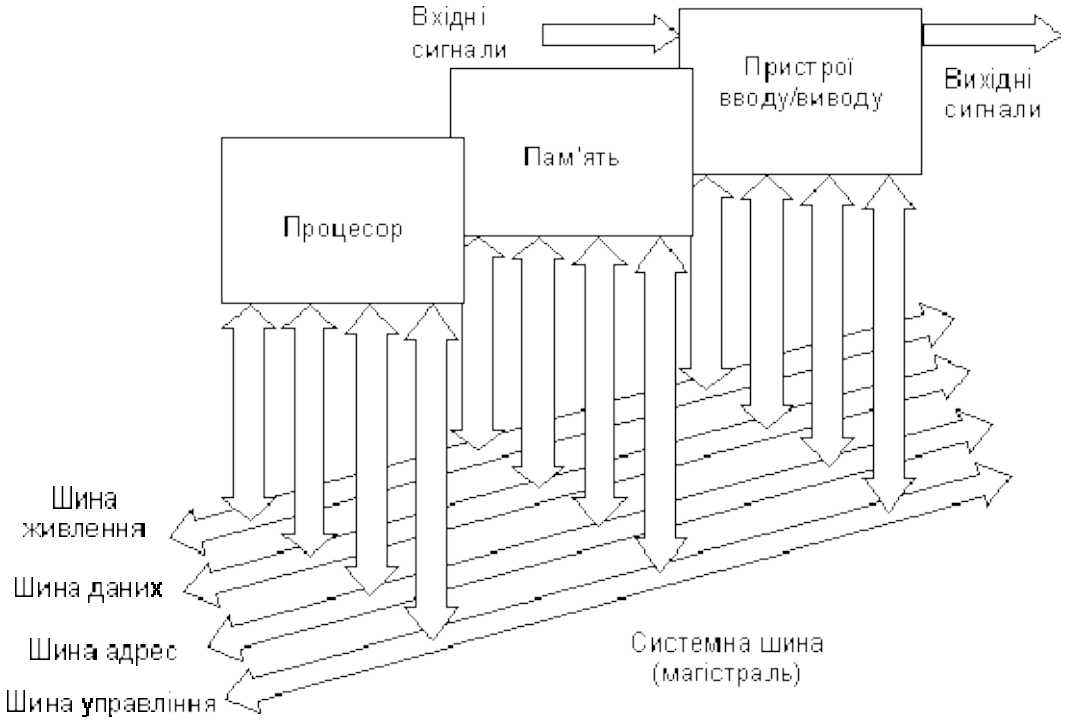


Рис. 5.10. Структура мікропроцесорної системи.

Усі пристрої мікропроцесорної системи поєднуються *загальною системною шиною* (вона ж називається ще *системною магістраллю* або *каналом*). Системна магістраль містить у собі чотири основні шини нижнього рівня:

- шина адреси (Address Bus);
- шина даних (Data Bus);
- шина управління (Control Bus);
- шина живлення (Power Bus).

Шина адреси служить для визначення адреси (номера) пристрою, з яким процесор обмінюється інформацією в даний момент. Кожному пристрою (крім процесора), кожній комірці пам'яті в мікропроцесорній системі присвоюється власна адреса. Коли код якоїсь адреси виставляється процесором на шині адреси, пристрій, якому ця адреса приписана, розуміє, що з ним зараз передбачається обмін інформацією. Шина адреси може бути однонапрямленою або двонапрямленою.

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

Шина управління, на відміну від шини адреси і шини даних, складається з окремих керуючих сигналів. Кожний з цих сигналів під час обміну інформацією має свою функцію. Деякі сигнали служать для стробу переданих або прийнятих даних (тобто визначають моменти часу, коли

інформаційний код виставлений на шину даних). Інші керуючі сигнали можуть використовуватися для підтвердження прийому даних, для скидання всіх пристроїв у вихідний стан (ініціалізації), для тактування всіх пристроїв і т.д. Лінії шини управління можуть бути однонапрямленими або двонапрямленими.

Нарешті, **шина живлення** призначена не для пересилання інформаційних сигналів, а для живлення системи. Вона складається з ліній живлення і загального провідника. У мікропроцесорній системі може бути одне джерело живлення (частіше +5 В) або кілька джерел живлення (переважно ще -5 В, +12 В та -12 В). Кожній напрузі живлення відповідає своя лінія зв'язку. Усі пристрої під'єднані до цих ліній паралельно.

Якщо в мікропроцесорну систему треба увести вхідний код (або вхідний сигнал), то процесор шиною адреси звертається до потрібного пристрою вводу/вводу і приймає шиною даних вхідну інформацію. Якщо з мікропроцесорної системи треба вивести вихідний код (або вихідний сигнал), то процесор звертається шиною адреси до потрібного пристрою вводу/вводу і передає йому шиною даних вихідну інформацію.

Якщо інформація повинна пройти складну багатоступінчасту обробку, то процесор може зберігати проміжні результати в системній оперативній пам'яті. Для звертання до будь-якої комірки пам'яті процесор виставляє її адресу на шину адреси і передає в неї інформаційний код шиною даних або ж приймає з неї інформаційний код шиною даних. У пам'яті (оперативній і постійній) знаходяться також і керуючі коди (команди виконуваної процесором програми), які процесор також читає шиною даних з адресацією шиною адреси. Постійна пам'ять використовується, в основному, для збереження програми початкового пуску мікропроцесорної системи, що виконується щораз після увімкнення живлення. Інформація в неї заноситься виробником раз і назавжди.

Таким чином, у мікропроцесорній системі всі інформаційні коди і коди команд передаються шинами послідовно, по черзі. Це визначає порівняно невисоку швидкодію мікропроцесорної системи. Воно обмежено зазвичай навіть не швидкодією процесора (яка теж є дуже важливою) і не швидкістю обміну системною шиною (магістраллю), а саме послідовним характером передачі інформації із системної шини (магістралі).

Важливо враховувати, що пристрої вводу/виводу найчастіше є пристроями на "жорсткій логіці". На них може бути покладена частина функцій, виконуваних мікропроцесорною системою. Тому в виробника завжди є можливість перерозподіляти функції системи між апаратною і програмною реалізаціями оптимальним чином. Апаратна реалізація прискорює виконання функцій, але має недостатню гнучкість. Програмна реалізація значно повільніша, але забезпечує високу гнучкість. Апаратна реалізація функцій збільшує вартість системи і її енергоспоживання, програмна - не збільшує. Найчастіше застосовується комбінування апаратних і програмних функцій.

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

5.3. Режими роботи мікропроцесорної системи

Як уже відзначалося, мікропроцесорна система забезпечує високу гнучкість роботи, вона здатна набудовуватися на будь-яку задачу. Гнучкість ця обумовлена насамперед тим, що функції, виконувані системою, визначаються програмою (програмним забезпеченням, software), яку виконує процесор. Апаратура (апаратне забезпечення, hardware) залишається незмінною при будь-якій задачі. Записуючи в пам'ять системи програму, можна змусити мікропроцесорну систему виконувати будь-яку задачу, що підтримується даною апаратурою. До того ж шинна організація зв'язків мікропроцесорної системи дозволяє досить легко заміняти апаратні модулі, наприклад, заміняти пам'ять на нову більшого об'єму або більш високої швидкодії, додавати або модернізувати пристрої вводу/виводу, нарешті, заміняти процесор на більш потужний. Це також дозволяє збільшити гнучкість системи, продовжити її життя при будь-якій зміні вимог до неї.

Але гнучкість мікропроцесорної системи визначається не тільки цим. Набудовуватися на задачу допомагає ще і вибір режиму роботи системи, тобто режиму обміну інформацією із системною магістраллю (шиною).

Практично будь-яка розвита мікропроцесорна система (у тому числі і комп'ютер) підтримує три основних режими обміну магістраллю:

- програмний обмін інформацією;
- обмін з використанням переривань (Interrupts);
- обмін з використанням прямого доступу до пам'яті (ПДП, DMA - Direct Memory Access).

Програмний обмін інформацією є основним у будь-якій мікропроцесорній системі. Він передбачений завжди, без нього неможливі інші режими обміну. У цьому режимі процесор є одноособовим хазяїном (або задатчиком, Master) системної магістралі. Всі операції (цикли) обміну інформацією в даному випадку ініціюються тільки процесором, усі вони виконуються строго в порядку, запропонованому програмою, яка виконується.

Процесор читає (вибирає) з пам'яті коди команд і виконує їх, читаючи дані з пам'яті або з пристрою вводу/виводу, обробляючи їх, записуючи дані в пам'ять або передаючи їх у пристрій вводу/виводу. Шлях процесора по програмі може бути лінійним, циклічним, може містити переходи (стрибки), але він завжди безупинний і цілком знаходиться під контролем процесора. Ні на які зовнішні події, не зв'язані з програмою, процесор не реагує (Рис. 1.11). Усі сигнали на магістралі в даному випадку контролюються процесором.

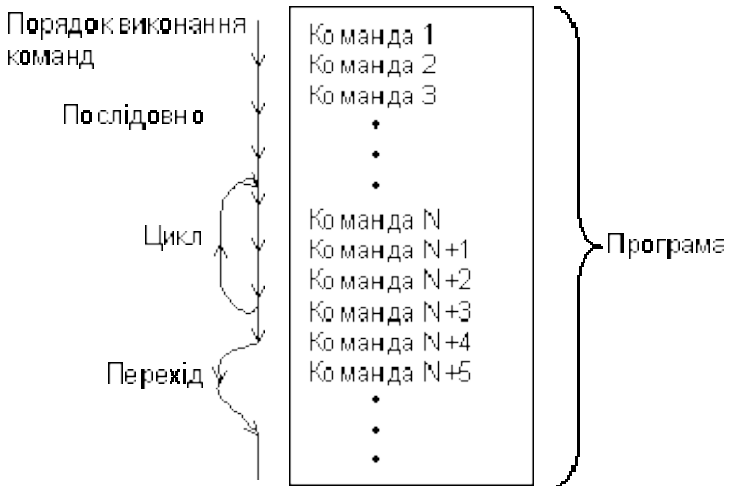


Рис. 5.11. Програмний обмін інформацією.

Обмін з використанням переривань використовується тоді, коли необхідна реакція мікропроцесорної системи на якусь зовнішню подію, на прихід зовнішнього сигналу. У випадку комп'ютера зовнішньою подією може бути, наприклад, натискання на клавіші клавіатури або прихід локальною мережею пакета даних. Комп'ютер повинен реагувати на це, відповідно, виведенням символу на екран або ж читання й обробкою прийнятого з мережі пакета.

У загальному випадку організувати реакцію на зовнішню подію можна трьома різними шляхами:

- за допомогою постійного програмного контролю факту настання події (так званий метод опитування прапора або polling);
- за допомогою переривання, тобто насильного переведення процесора з виконання поточної програми на виконання негайно необхідної програми;
- за допомогою прямого доступу до пам'яті, тобто без участі процесора при його відключенні від системної магістралі.

Проілюструвати ці три способи можна наступним простим прикладом. Допустимо, ви готуєте собі сніданок, поставивши на плиту кип'ятитися молоко. Природно, на закипання молока треба реагувати, причому терміново. Як це організувати? Перший шлях - постійно стежити за молоком, але тоді ви нічого іншого робити не зможете. Вірніше буде регулярно поглядати на молоко, роблячи одночасно щось інше. Це програмний режим з опитуванням прапора. Другий шлях - установити на каструлю з молоком датчик, що подасть звуковий сигнал при закипанні молока, і спокійно займатися іншими справами. Почувши сигнал, ви знімете з плити молоко. Правда, можливо, вам доведеться спочатку закінчити те, що ви почали робити, так що ваша реакція буде повільнішою, ніж у першому випадку. Нарешті, третій шлях полягає в тому, щоб з'єднати датчик на каструлі з керуванням плитою так, щоб при закипанні молока пальник був вимкнений без вашої участі (правда, аналогія з ПДП тут не дуже точна, тому що в даному випадку на момент виконання дії вас не відволікають від роботи).

Перший випадок з опитуванням прапора реалізується в мікропроцесорній системі постійним читанням інформації процесором із пристрою вводу/виводу, який зв'язаний з тим зовнішнім пристроєм, на поведінку якого необхідно терміново реагувати.

В другому випадку в режимі переривання процесор, отримавши запит переривання від зовнішнього пристрою (часто званий IRQ - Interrupt ReQuest), закінчує виконання поточної команди і переходить до програми обробки переривання. Закінчивши виконання програми обробки переривання, він повертається до перерваної програми з того місця, де його перервали (Рис. 1.12).

Тут важливим є те, що вся робота, як і у випадку програмного режиму, здійснюється самим процесором, зовнішня подія просто тимчасово відволікає його від виконання програми. Реакція на зовнішню подію по перериванню, в загальному випадку, повільніша, ніж при програмному режимі. Як і у випадку програмного обміну, тут усі сигнали на магістралі виставляються процесором, тобто він цілком контролює магістраль. Для обслуговування переривань у систему іноді вводиться спеціальний модуль контролера переривань, але він в обміні інформацією участі не бере. Його завдання полягає в тому, щоб спростити роботу процесора з зовнішніми запитами переривань. Цей контролер, зазвичай, програмно керується процесором системною магістраллю.

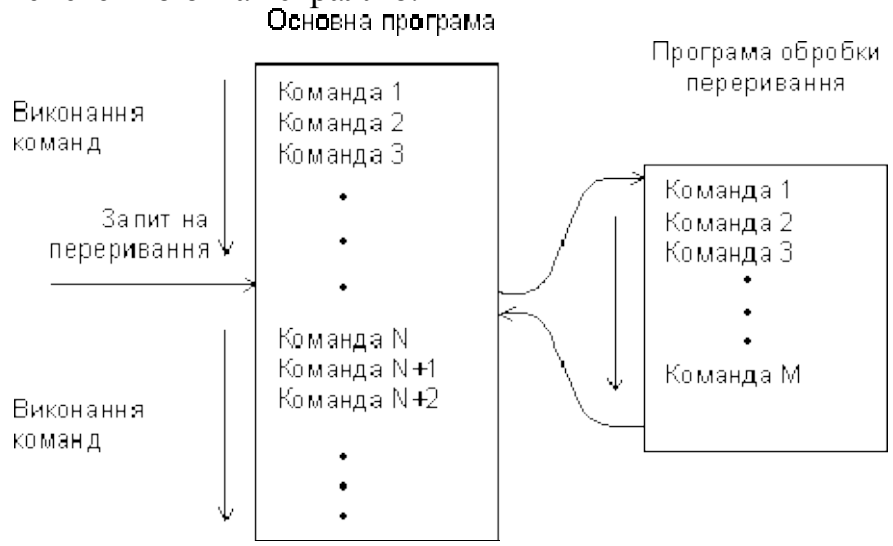


Рис. 5.12. Обслуговування переривання.

Природно, ніякого прискорення роботи системи переривання не дає. Його застосування дозволяє тільки відмовитися від постійного опитування прапора зовнішньої події і тимчасово, до її настання, зайняти процесор виконанням якихось інших задач.

Прямий доступ до пам'яті (ПДП, DMA) - це режим, що принципово відрізняється від двох раніше розглянутих режимів тим, що обмін системною шиною йде без участі процесора. Зовнішній пристрій, що вимагає обслуговування, сигналізує процесору, що режим ПДП необхідний, у відповідь на це процесор закінчує виконання поточної команди і

відключається від усіх шин, сигналізуючи пристрою, який подав запит, що обмін у режимі ПДП можна починати.

Операція ПДП зводиться до пересилання інформації з пристрою вводу/виводу в пам'ять або з пам'яті в пристрій вводу/виводу. Коли пересилання інформації буде завершено, процесор знову повертається до перерваної програми, продовжуючи її з того місця, де його перервали (Рис. 1.13). Це схоже на режим обслуговування переривань, але в даному випадку процесор не бере участі в обміні. Як і у випадку переривань, реакція на зовнішню подію при ПДП істотно повільніша, ніж при програмному режимі.

Зрозуміло, що в цьому випадку потрібно введення в систему додаткового пристрою (контролера ПДП), що буде здійснювати повноцінний обмін системною магістраллю без всякої участі процесора. Причому процесор попередньо повинен повідомити цьому контролеру ПДП, звідкіля йому варто брати інформацію і/або куди її потрібно поміщати. Контролер ПДП може вважатися спеціалізованим процесором, який відрізняється тим, що сам не бере участі в обміні, не приймає в себе інформацію і не видає її (Рис. 1.14).

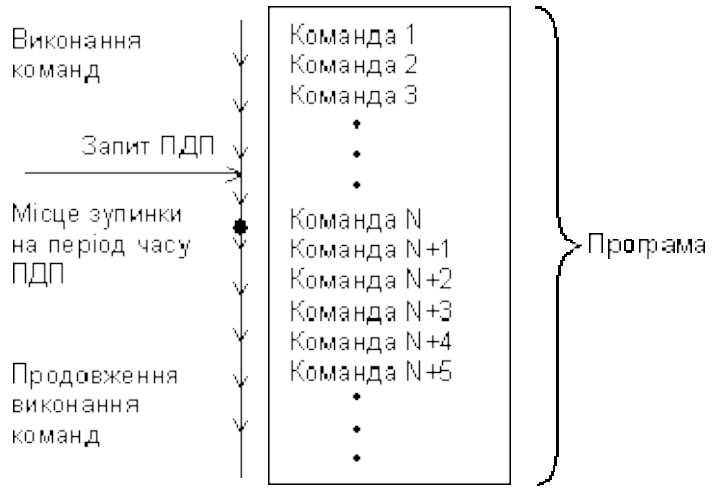


Рис. 5.13. Обслуговування ПДП.

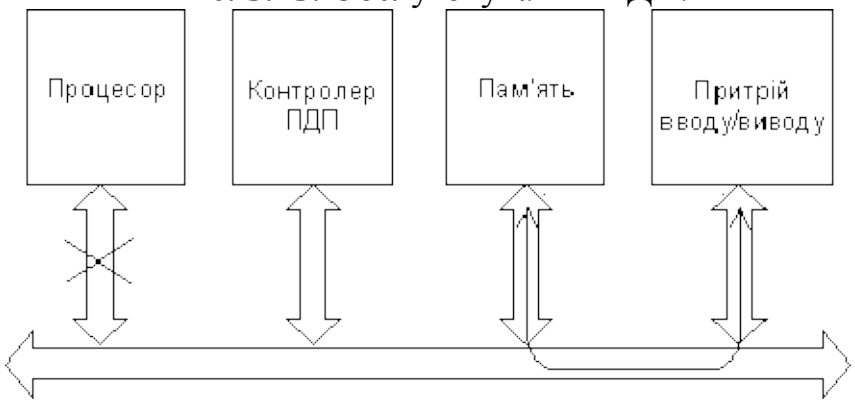


Рис. 5.14. Інформаційні потоки в режимі ПДП.

У принципі, контролер ПДП може входити до складу пристрою вводу/виводу, якому потрібен режим ПДП або, навіть, до складу декількох пристроїв вводу/виводу.

Теоретично обмін за допомогою прямого доступу до пам'яті може забезпечити більш високу швидкість передачі інформації, аніж програмний обмін, тому що процесор передає дані повільніше, ніж спеціалізований контролер ПДП. Однак на практиці ця перевага реалізується далеко не завжди. Швидкість обміну в режимі ПДП зазвичай обмежена можливостями магістралі. До того ж необхідність програмного задавання режимів контролера ПДП може звести нанівець виграш від більш високої швидкості пересилання даних у режимі ПДП. Тому режим ПДП застосовується рідко.

Якщо в системі вже існує самостійний контролер ПДП, то це може в ряді випадків істотно спростити апаратуру пристроїв вводу/виводу, які працюють у режимі ПДП. У цьому, мабуть, полягає єдина безперечна перевага режиму ПДП.

5.4. Архітектура мікропроцесорних систем

Дотепер ми розглядали тільки один тип **архітектури** мікропроцесорних систем - архітектуру з загальною, єдиною шиною для даних і команд (одношинну, або *принстонську*,*фон-нейманівську* архітектуру). Відповідно, у складі системи в цьому випадку присутня одна загальна пам'ять, як для даних, так і для команд (Рис. 1.15).

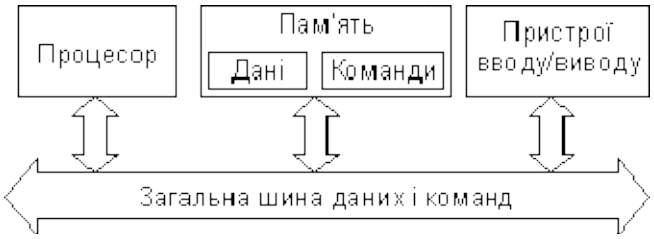


Рис. 5.15. Архітектура з загальною шиною даних і команд.

Але існує також і альтернативний тип архітектури мікропроцесорної системи - це архітектура з роздільними шинами даних і команд (двохшинна, або *гарвардська* архітектура). Ця архітектура припускає наявність у системі окремої пам'яті для даних і окремої пам'яті для команд (Рис. 1.16). Обмін процесора з кожним із двох типів пам'яті відбувається по своїй шині.

Архітектура з загальною шиною поширена набагато більше, вона застосовується, наприклад, у персональних комп'ютерах і в складних мікрокомп'ютерах. Архітектура з роздільними шинами застосовується, в основному, в однокристальних мікроконтролерах.

Розглянемо деякі переваги і недоліки обидвох архітектурних рішень.

Архітектура з загальною шиною (принстонська, фон-нейманівська) є простішою, вона не вимагає від процесора одночасного обслуговування двох шин, контролю обміну двома шинами відразу. Наявність єдиної пам'яті даних і команд дозволяє гнучко розподіляти її об'єм між кодами даних і команд. Наприклад, у деяких випадках потрібна велика і складна програма, а даних у пам'яті треба зберігати не надто багато. В інших випадках, навпаки, програма проста, але необхідні великі об'єми збережених даних. Перерозподіл пам'яті не викликає ніяких проблем, головне - щоб програма і дані разом

вміщувалися в пам'яті системи. Як правило, у системах з такою архітектурою пам'ять буває досить великого об'єму (до десятків і сотень мегабайт). Це дозволяє вирішувати самі складні задачі.

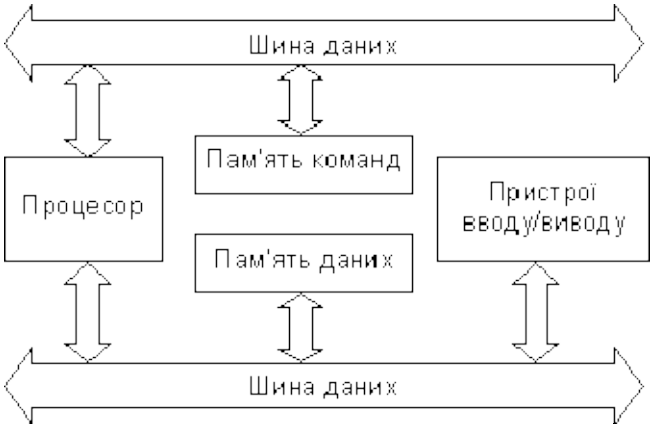


Рис. 5.16. Архітектура з роздільними шинами даних і команд.

Архітектура з роздільними шинами даних і команд складніша, вона змушує процесор працювати одночасно з двома потоками кодів, обслуговувати обмін двома шинами одночасно. Програма може розміщуватися тільки в пам'яті команд, дані - тільки в пам'яті даних. Така вузька спеціалізація обмежує коло задач, які розв'язуються системою, тому що не дає можливості гнучкого перерозподілу пам'яті. Пам'ять даних і пам'ять команд у цьому випадку мають порівняно невеликий об'єм, тому застосування систем з даною архітектурою обмежується переважно не надто складними задачами.

У чому ж перевага архітектури з двома шинами (гарвардської)? У першу чергу, у швидкодії.

Справа в тому, що при єдиній шині команд і даних процесор змушений по одній цій шині приймати дані (з пам'яті або пристрою вводу/виводу) і передавати дані (у пам'ять або в пристрій вводу/виводу), а також читати команди з пам'яті. Природно, одночасно ці пересилання кодів магістралями відбуватися не можуть, вони повинні відбуватися по черзі. Сучасні процесори здатні поєднати в часі виконання команд і проведення циклів обміну системною шиною. Використання конвеєрних технологій і швидкої кеш-пам'яті дозволяє їм прискорити процес взаємодії з порівняно повільною системною пам'яттю. Підвищення тактової частоти й удосконалення структури процесорів дають можливість скоротити час виконання команд. Але подальше збільшення швидкодії системи можливе тільки при поєднанні пересилання даних і читання команд, тобто при переході до архітектури з двома шинами.

У випадку двохшинної архітектури обмін обидвома шинами може бути незалежним, паралельним у часі. Відповідно, структури шин (кількість розрядів коду адреси і коду даних, порядок і швидкість обміну інформацією і т.д.) можуть бути обрані оптимально для тієї задачі, яка виконується кожною шиною. Тому за інших рівних умов перехід на двохшинну архітектуру прискорює роботу мікропроцесорної системи, хоча і вимагає додаткових

витрат на апаратуру, ускладнення структури процесора. Пам'ять даних у цьому випадку має свій розподіл адрес, а пам'ять команд - свій.

Найпростіше переваги двохшинної архітектури реалізуються усередині однієї мікросхеми. У цьому випадку можна також істотно зменшити вплив недоліків цієї архітектури. Тому основне її застосування - у мікроконтролерах, від яких не потрібно рішення надто складних задач, але зате необхідна максимальна швидкодія при заданій тактовій частоті.

5.5. Типи мікропроцесорних систем

Діапазон застосування мікропроцесорної техніки зараз дуже широкий, вимоги до мікропроцесорних систем пред'являються самі різні. Тому сформувалося кілька типів мікропроцесорних систем, що відрізняються потужністю, універсальністю, швидкодією і структурними відмінностями. Основні типи наступні:

- **мікроконтролери** - найпростіший тип мікропроцесорних систем, у яких усі або більшість вузлів системи виконані у виді однієї мікросхеми;
- **контролери** - керуючі мікропроцесорні системи, що виконані у виді окремих модулів;
- **мікрокомп'ютери** - потужніші мікропроцесорні системи з розвинутими засобами сполучення з зовнішніми пристроями.
- **комп'ютери** (у тому числі персональні) - самі потужні і найуніверсальніші мікропроцесорні системи.

Чітку межу між цими типами іноді провести досить складно. Швидкодія всіх типів мікропроцесорів постійно зростає, і нерідкі випадки, коли новий мікроконтролер виявляється швидшим, наприклад, персонального комп'ютера, який застарів. Але деякі принципові відмінності все-таки існують.

Мікроконтролери є універсальними пристроями, які практично завжди використовуються не самі по собі, а в складі більш складних пристроїв, у тому числі і контролерів. Системна шина мікроконтролера схована від користувача усередині мікросхеми. Можливості підключення зовнішніх пристроїв до мікроконтролера обмежені. Пристрої на мікроконтролерах зазвичай призначені для рішення однієї задачі.

Контролери, як правило, створюються для рішення якоїсь окремої задачі або групи близьких задач. Вони звичайно не мають можливостей підключення додаткових вузлів і пристроїв, наприклад, великої пам'яті, засобів вводу/виводу. Їхня системна шина найчастіше недоступна користувачеві. Структура контролера проста й оптимізована під максимальну швидкодію. У більшості випадків виконувані програми зберігаються в постійній пам'яті і не змінюються. Конструктивно контролери випускаються в одноплатному варіанті.

Мікрокомп'ютери відрізняються від контролерів більш відкритою структурою, вони допускають підключення до системної шини декількох додаткових пристроїв. Виробляються мікрокомп'ютери в каркасі, корпусі з роз'ємами системної магістралі, доступними користувачеві.

Мікрокомп'ютери можуть мати засоби збереження інформації на магнітних носіях (наприклад, магнітні диски) і досить розвинуті засоби зв'язку з користувачем (відеомонітор, клавіатура). Мікрокомп'ютери розраховані на широке коло задач, але на відміну від контролерів, до кожної нової задачі його треба пристосовувати заново. Виконувані мікрокомп'ютером програми можна легко змінювати.

Нарешті, **комп'ютери** і самі розповсюджені з них - **персональні комп'ютери** - це самі універсальні з мікропроцесорних систем. Вони обов'язково передбачають можливість модернізації, а також широкі можливості підключення нових пристроїв. Їхня системна шина, зазвичай, доступна користувачеві. Крім того, зовнішні пристрої можуть підключатися до комп'ютера через кілька вмонтованих портів зв'язку (кількість портів доходить іноді до 10). Комп'ютер завжди має сильно розвиті засоби зв'язку з користувачем, засоби тривалого збереження інформації великого об'єму, засоби зв'язку з іншими комп'ютерами інформаційними мережами. Області застосування комп'ютерів можуть бути самими різними: математичні розрахунки, обслуговування доступу до баз даних, керування роботою складних електронних систем, комп'ютерні ігри, підготовка документів і т.д.

Будь-яку задачу, в принципі, можна виконати за допомогою кожного з перерахованих типів мікропроцесорних систем. Але при виборі типу треба по можливості уникати надмірності і передбачати необхідну для даної задачі гнучкість системи.

В даний час при розробці нових мікропроцесорних систем найчастіше вибирають шлях використання мікроконтролерів (приблизно в 80% випадків). При цьому мікроконтролери застосовуються або самостійно, з мінімальною додатковою апаратурою, або в складі більш складних контролерів з розвинутими засобами вводу/виводу.

Класичні мікропроцесорні системи на базі мікросхем процесорів і мікропроцесорних комплектів випускаються зараз досить рідко, у першу чергу, через складність процесу розробки і налагодження цих систем. Даний тип мікропроцесорних систем вибирають, в основному, тоді, коли мікроконтролери не можуть забезпечити необхідних характеристик.

Нарешті, помітне місце займають зараз мікропроцесорні системи на основі персонального комп'ютера. Виробникові в цьому випадку потрібно тільки оснастити персональний комп'ютер додатковими пристроями сполучення, а ядро мікропроцесорної системи уже є готовим. Персональний комп'ютер має розвинуті засоби програмування, що істотно спрощує задачу виробника. До того ж він може забезпечити самі складні алгоритми обробки інформації. Основні недоліки персонального комп'ютера - великі розміри корпусу й апаратурна надмірність для простих задач. Недоліком є і непристосованість більшості персональних комп'ютерів до роботи в складних умовах (запиленість, висока вологість, вібрації, високі температури і т.д.). Однак випускаються і спеціальні персональні комп'ютери, які пристосовані до різних умов експлуатації.

5.6. Регістри процесора

Як уже згадувалося, внутрішні регістри процесора представляють собою надоперативну пам'ять невеликого розміру, що призначена для тимчасового збереження службової інформації чи даних. Кількість регістрів у різних процесорах може бути від 6-8 до декількох десятків. Регістри можуть бути універсальними і спеціалізованими. Спеціалізовані регістри, що присутні в більшості процесорів, - це регістр-лічильник команд, регістр стану (**PSW**), регістр покажчика стека. Інші регістри процесора можуть бути як універсальними, так і спеціалізованими.

Наприклад, у 16-розрядному процесорі Т-11 фірми DEC було 8 регістрів загального призначення (РЗП) і один регістр стану. Усі регістри мали по 16 розрядів. З регістрів загального призначення один виділявся під лічильник команд, інший - під показник стека. Всі інші регістри загального призначення цілком взаємозамінні, тобто мають універсальне призначення, можуть зберігати як дані, так і адреси (покажчики), індекси і т.д. Максимально допустимий об'єм пам'яті для даного процесора складав 64 Кбайт (адрес пам'яті 16-розрядний).

У 16-розрядному процесорі MC68000 фірми Motorola було 19 регістрів: 16-розрядний регістр стану, 32-розрядний регістр **лічильника команд**, 9 регістрів адреси (32-розрядних) і 8 регістрів даних (32-розрядних). Два регістри адреси відведені під показники стека. Максимально допустимий об'єм адресованої пам'яті - 16 Мбайт (зовнішня шина адреси 24-розрядна). Усі 8 регістрів даних взаємозамінні. 7 регістрів адреси - теж взаємозамінні.

У 16-розрядному процесорі Intel 8086, що став базовим у лінії процесорів, які використовувались у персональних комп'ютерах, реалізований принципово інший підхід. Кожен регістр цього процесора має своє особливе призначення, і заміняти один одного регістри можуть тільки частково чи ж не можуть узагалі. Зупинимося на особливостях цього процесора докладніше.

Процесор 8086 має 14 регістрів розрядністю по 16 біт. З них чотири регістри (AX, BX, CX, DX) - це регістри даних, кожен з яких крім збереження операндів і результатів операцій має ще і своє специфічне призначення:

- *регістр AX* - множення, ділення, обмін із пристроями вводу/виводу (команди вводу і виводу);
- *регістр BX* - базовий регістр в обчисленнях адреси;
- *регістр CX* - лічильник циклів;
- *регістр DX* - визначення адреси вводу/виводу.

Для регістрів даних існує можливість роздільного використання обидвох байтів (наприклад, для регістра AX вони мають позначення AL - молодший байт і AH - старший байт).

Наступні чотири внутрішніх регістри процесора - це сегментні регістри, кожний з яких визначає положення одного з робочих сегментів (Рис. 3.10):

- *регістр CS* (Code Segment) відповідає сегменту команд, які виконуються в даний момент;

- *регістр DS* (Data Segment) відповідає сегменту даних, з якими працює процесор;
- *регістр ES* (Extra Segment) відповідає додатковому сегменту даних;
- *регістр SS* (Stack Segment) відповідає сегменту стека.

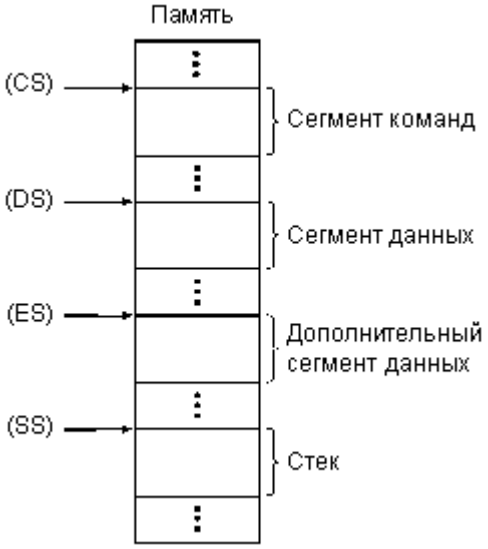


Рис. 5.17. Сегменти команд, даних і стека в пам'яті.

У принципі, усі ці сегменти можуть і перекриватися для оптимального використання простору пам'яті. Наприклад, якщо програма займає тільки частину сегмента, то сегмент даних може починатися відразу після завершення роботи програми (з точністю 16 байт), а не після закінчення всього сегмента програми.

Наступні п'ять регістрів процесора (SP - Stack Pointer, BP - Base Pointer, SI - Source Index, DI - Destination Index, IP -Instruction Pointer) служать показниками (тобто визначають зсув у межах сегмента). Наприклад, лічильник команд процесора утвориться парою регістрів CS і IP, а покажчик стека - парою регістрів SP і SS. Регістри SI, DI використовуються в лінійкових операціях, тобто при послідовній обробці декількох комірок пам'яті однією командою.

Останній регістр **FLAGS** - це регістр стану процесора (**PSW**). З його 16 розрядів використовуються тільки дев'ять (Рис. 3.11): CF (Carry Flag) - прапор переносу при арифметичних операціях, PF (Parity Flag) - прапор парності результату, AF (Auxiliary Flag) - прапор додаткового переносу, ZF (Zero Flag) - прапор нульового результату, SF (Sign Flag) - прапор знаку (збігається зі старшим бітом результату), TF (Trap Flag) - прапор покрокового режиму (використовується при налагодженні), IF (Interrupt-enable Flag) - прапор дозволу апаратних переривань, DF (Direction Flag) - прапор напрямку при лінійкових операціях, OF (Overflow Flag) - прапор переповнення.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Рис. 5.18. Регістр стану процесора 8086.

Біти регістра стану встановлюються чи очищаються в залежності від результату виконання попередньої команди і використовуються деякими командами процесора. Біти регістра стану можуть також встановлюватися й очищатися спеціальними командами процесора (про систему команд процесора буде розказано в наступному розділі).

У багатьох процесорах виділяється спеціальний регістр, який називається *акумулятором* (тобто накопичувачем). При цьому, як правило, тільки цей регістр-акумулятор може брати участь у всіх операціях, тільки через нього може відбуватися взаємодія з пристроями вводу/виводу. Іноді в ньому ж міститься результат будь-якої виконаної команди (у цьому випадку говорять навіть про "акумуляторну" архітектуру процесора). Наприклад, у процесорі 8086 регістр даних AX можна вважати своєрідним акумулятором, тому що саме він обов'язково бере участь у командах множення і ділення, а також тільки через нього можна пересилати дані в пристрої вводу/виводу і з них. Виділення спеціального регістра-акумулятора спрощує структуру процесора і прискорює пересилання кодів усередині процесора, але в деяких випадках сповільнює роботу системи в цілому, тому що весь потік інформації повинен пройти через один регістр-акумулятор. У випадку, коли кілька регістрів процесора цілком взаємозамінні, таких проблем не виникає.

5.7. Лічильник команд

Лічильник команд PCL і PCLATH має розрядність 13 біт. Молодший байт лічильника (PCL) доступний для читання і запису і знаходиться в регістрі 02h. Старший байт лічильника команд не може бути прямо записаний або зчитаний і береться з регістра PCLATH (PC latch high), адреса якого 0Ah. Уміст PCLATH передається в старший байт лічильника команд, коли він завантажується новим значенням.

У залежності від того, чи завантажується в лічильник команд нове значення під час виконання команд CALL, GOTO, або в молодший байт лічильника команд (PCL) відбувається запис, - старші біти лічильника команд завантажуються з PCLATH різними способами, як показано на Рис. 5.6.

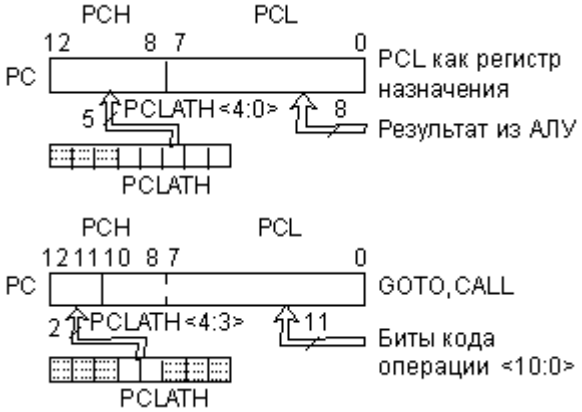


Рис. 5.19. Завантаження старших бітів лічильника команд.

Команди CALL і GOTO оперують 11-розрядним адресним діапазоном, достатнім для зсуву в межах сторінки програмної пам'яті об'ємом 2К слів.

Для МК підгрупи PIC16F8X цього вистачає. З метою забезпечення можливості розширення пам'яті команд для майбутніх моделей МК передбачене завантаження двох старших біт лічильника команд із регістра PCLATH<4:3>. При використанні команд CALL і GOTO користувач повинен переконатися в тому, що ці сторінкові біти запрограмовані для виходу на потрібну сторінку. При виконанні команди CALL або виконанні переривання весь 13-бітний лічильник команд поміщується в стек, тому для повернення з підпрограми не потрібні маніпуляції з розрядами PCLATH<4:3>.

Мікроконтролери підгрупи PIC16F8X ігнорують значення біт PCLATH<4:3≦, які використовуються для звертання до сторінок 1, 2 і 3 програмної пам'яті. Однак застосовувати біти PCLATH<4:3> як комірки пам'яті загального призначення не рекомендується, тому що це може вплинути на сумісність з майбутніми поколіннями виробів.

Можливість виконувати арифметичні операції безпосередньо над лічильником команд дозволяє дуже швидко й ефективно здійснювати табличні перетворення в PIC-контролерах.

Мікроконтролери підгрупи PIC16F8X мають восьмирівневий апаратний стек шириною 13 біт (див. Рис. 5.4). Область стека не належить ні до програмної області, ні до області даних, а покажчик стека користувачу недоступний. Поточне значення лічильника команд посилається в стек, коли виконується команда CALL або відбувається обробка переривання. При виконанні процедури повернення з підпрограми (команди RETLW, RETFIE або RETURN) уміст лічильника команд відновлюється зі стека. Регістр PCLATH при операціях зі стеком не змінюється.

Стек працює як циклічний буфер. Отже, після того як стек був завантажений 8 разів, дев'яте завантаження перепише значення першої. Десяте завантаження перепише другу і т.д. Якщо стек був вивантажений 9 разів, лічильник команд стає таким ж, як після першого вивантаження.

Ознак положення стека в контролері не передбачено, тому користувач повинен самостійно стежити за рівнем вкладення підпрограм.

5.8. Пряма і непряма адресації

Коли відбувається пряма 9-бітна адресація, молодший 7 біт береться як пряма адреса з коду операції, а два біти покажчика сторінок (RP1, RP0) з регістра статусу, як показано на Рис. 5.7.

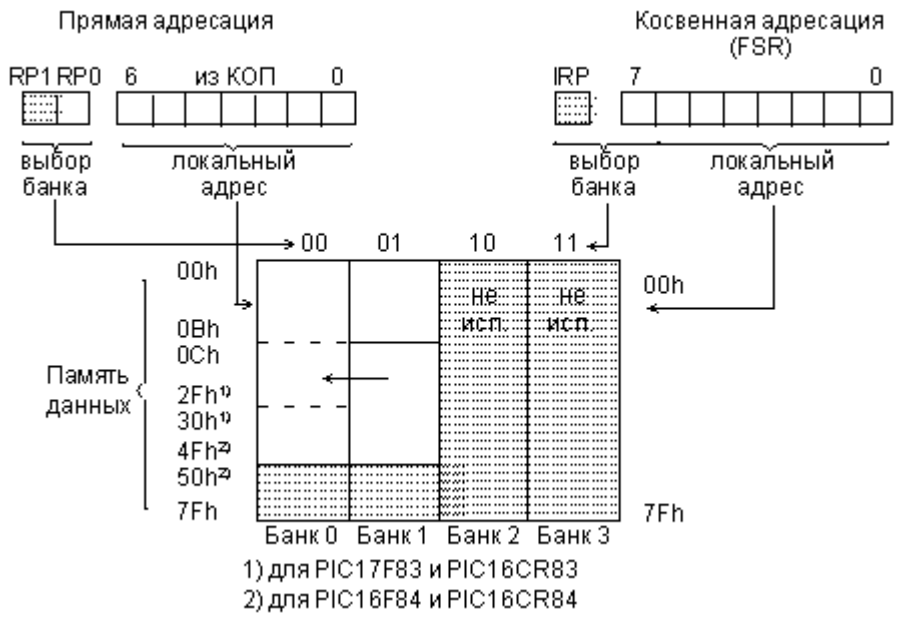


Рис. 5.20. Методи адресації даних.

Ознакою непрямої адресації є звертання до регістру INDF. Будь-яка команда, що використовує INDF (адресу 00h) як регістр фактично звертається до покажчика, що зберігається в FSR (адреса 04h). Читання непрямим чином самого регістра INDF дасть результат 00h. Запис у регістр INDF непрямим чином буде виглядати як NOP, але біти статусу можуть бути змінені. Необхідна 9-бітна адреса формується шляхом об'єднання умісту 8-бітного FSR регістра і біта IRP з регістра статусу (див. Рис. 5.7).

Зверніть увагу, що деякі регістри спеціальних функцій розташовуються в банку 1. Щоб адресуватися до них, потрібно додатково установити в одиницю біт RP0 у регістрі статусу.

5.9. Порти вводу/виводу

Контролери підгрупи PIC16F8X мають два порти: PORTA (5 біт) і PORTB (8 біт) з побітовим індивідуальним настроюванням на ввід або на вивід.

Порт A (PORTA) являє собою 5-бітовий фіксатор, що відповідає виводам контролера RA. Лінія RA4 має вхід тригера Шмідта і вихід з відкритим стоком. Всі інші лінії порту мають ТТЛ вхідні рівні і КМОН вихідні буфери. Адреса регістра порту A - 05h.

Кожній лінії порту поставлений у відповідність біт напрямку передачі даних, що зберігається в керуючому регістрі TRISA, розташованому за адресою 85h. Якщо біт керуючого TRISA регістра має значення 1, то відповідна лінія буде встановлюватися на ввід. Нуль перемикає лінію на вивід і одночасно виводить на неї уміст відповідного регістра-фіксатора порту. При увімкненні живлення всі лінії порту за замовчуванням настроюються на ввід.

На Рис. 5.8 дано схему ліній RA<3:0> порту A.

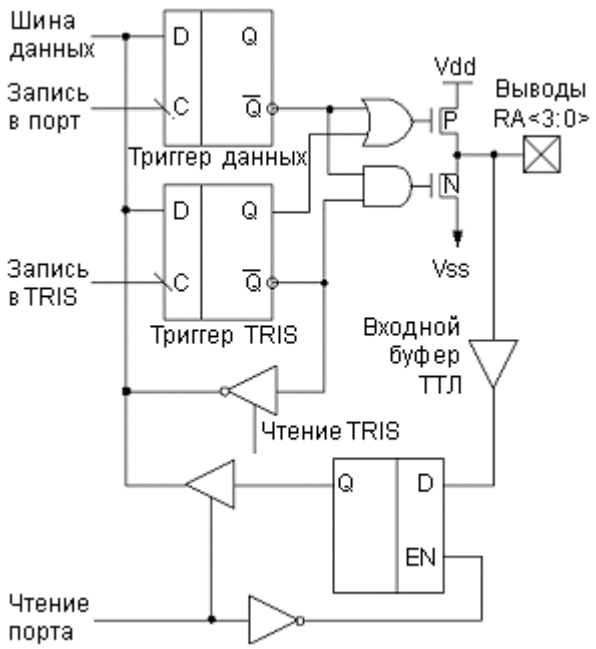


Рис. 5.21. Схема ліній RA<3:0> порту А. Виводи порту мають захисні діоди до Vdd і Vss.

Операція читання порту А зчитує стан виводів порту, у той час як запис у нього змінює стан тригерів порту. Всі операції з портом є операціями типу "читання-модифікація-запис". Тому запис у порт припускає, що стан виводів порту спочатку зчитується, потім модифікується і записується в тригер-фіксатор.

Вивід RA4 мультиплексований з тактовим входом таймера TMR0. Схема лінії RA4 порта А наведена на Рис. 5.9.

Порт В (PORTB) - це двонапрямлений 8-бітовий порт, що відповідає виводам RB<7:0> контролера і розташований за адресою 06h. Керуючий регістр, який відноситься до цього порту TRISB розташований на першій сторінці регістрів за адресою 86h. Якщо біт керуючого TRISB регістра має значення 1, то відповідна лінія буде встановлюватися на ввід. Нуль перемикає лінію на вивід і одночасно виводить на неї уміст відповідного регістра. При увімкненні живлення всі лінії порту за замовчуванням настроюються на ввід.

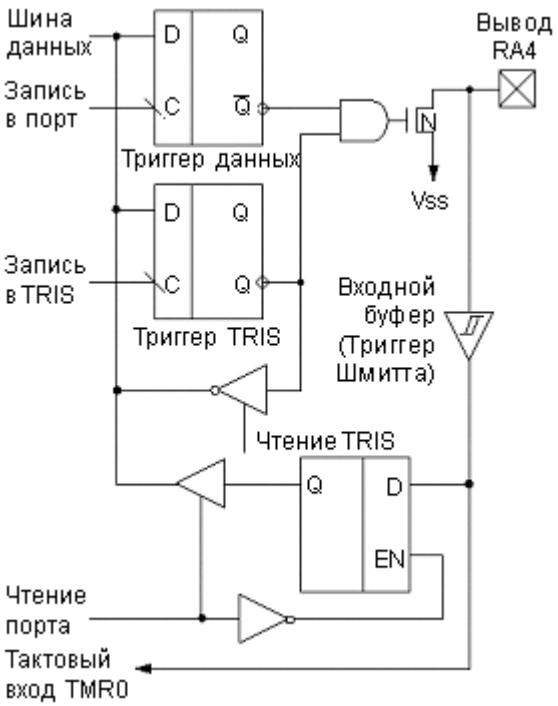


Рис. 5.22. Схема лінії RA4 порти A. Вивід порту має захисний діод тільки до Vss.

У кожної ніжки порту B є невелике активне навантаження (близько 100мкА) на лінію живлення (pull-up). Вона автоматично відключається, якщо ця ніжка запрограмована як вивід. Більше того, керуючий біт /RBPU регістра OPTION<7> може відключити (при RBPU=1) усі навантаження. Ініціалізація при увімкненні живлення також відключає всі навантаження.

Чотири лінії порту B (RB<7:4>) можуть викликати переривання при зміні значення сигналу на кожній з них. Якщо ці лінії настроєні на ввід, то вони опитуються і замикаються в циклі читання Q1. Нова величина вхідного сигналу порівнюється з старою в кожному командному циклі. При розбіжності значення сигналу на ніжці й у фіксаторі генерується високий рівень. Виходи детекторів "розбіжностей" RB4, RB5, RB6, RB7 з'єднуються за схемою АБО і генерують переривання RBIF (запам'ятовується в регістрі INTCON<0>). Будь-яка лінія, настроєна як вивід, у цьому порівнянні участі не бере. Переривання може вивести кристал з режиму SLEEP. У підпрограмі обробки переривання варто скинути запит переривання одним з наступних способів:

- прочитати (або записати в) порт B. Це завершить стан порівняння;
- обнулити біт RBIF регістра INTCON <0>.

При цьому необхідно мати на увазі, що умова "розбіжності" буде продовжувати встановлювати ознаку RBIF. Тільки читання порту B може усунути "розбіжність" і дозволить обнулити біт RBIF.

Переривання по розбіжності і програмно установлювані внутрішні активні навантаження на цих чотирьох лініях можуть забезпечити простий інтерфейс, наприклад, із клавіатурою, з виходом з режиму SLEEP по натисканню клавіш.

Схеми ліній порту B наведені на Рис. 5.10 і 5.11.

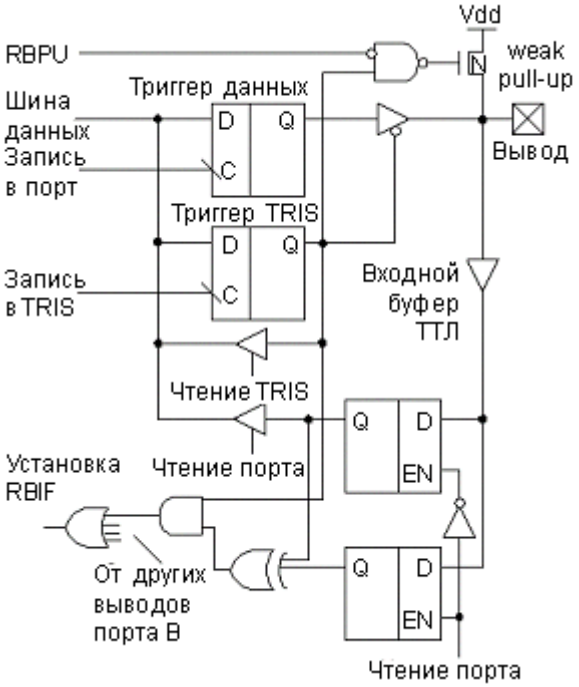


Рис. 5.23. Схема ліній RB<7:4> порту B. Виводи порту мають захисні діоди до Vdd і Vss.

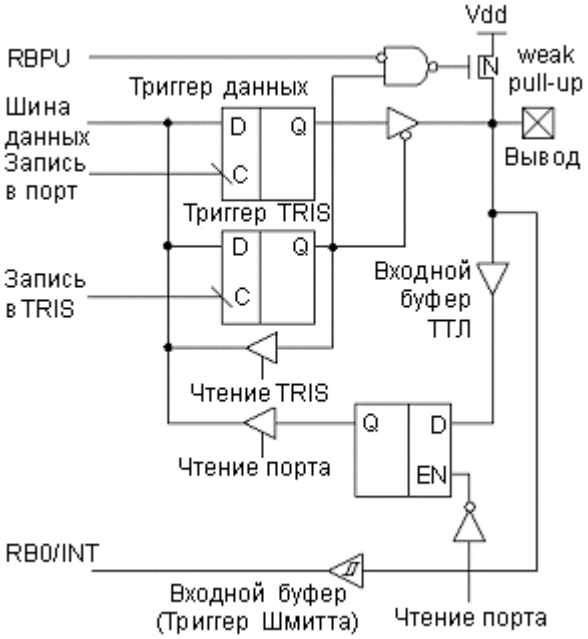


Рис. 5.24. Схема ліній RB<3:0> порту B. Виводи порту мають захисні діоди до Vdd і Vss.

При організації двонапрямлених портів необхідно враховувати особливості організації вводу/виводу даних МК. Будь-яка команда, що здійснює запис, виконує її усередині як "читання-модифікація-запис". Наприклад, команди BCF і BSF зчитують порт цілком, модифікують один біт і виводять результат назад. Тут необхідна обережність. Зокрема, команда BSF PORTB, 5 (установити в одиницю біт 5 порту B) спочатку зчитує всі реальні значення сигналів, що є присутніми у даний момент на виводах

порту. Потім виконуються дії над бітом 5, і нове значення байта цілком записується у вихідні фіксатори. Якщо інший біт регістра PORTB використовується як двонапрямлений порт вводу/виводу (скажемо, біт 0), і в даний момент він визначений як вхідний, то вхідний сигнал на цьому виводі буде зчитаний і записаний назад у вихідний тригер-фіксатор цього ж виводу, стираючи попередній стан. Доти, поки ця ніжка залишається в режимі вводу, ніяких проблем не виникає. Однак якщо пізніше лінія 0 переключиться в режим виводу, її стан буде невизначеним.

На ніжку, що працює в режимі виводу, не повинні навантажуватися зовнішні джерела струмів ("монтажне І", "монтажне АБО"). Великі результуючі струми можуть пошкодити кристал.

Необхідно витримувати визначену послідовність звертання до портів вводу/виводу. Запис у порт виводу відбувається наприкінці командного циклу. Але при читанні дані повинні бути стабільні на початку командного циклу. Будьте уважні в операціях читання, що слідують відразу за записом у той же порт. Тут треба враховувати інерційність встановлення напруги на виводах. Може знадобитися програмна затримка, щоб напруга на ніжці (яка залежить від навантаження) встигла стабілізуватися до початку виконання наступної команди читання.

ТЕМА 6. КОМАНДИ МІКРОПРОЦЕСОРА

Процесор (Рис. 6.1) звичайно являє собою окрему чи мікросхему ж частина мікросхеми (у випадку мікроконтролера). В колишні роки процесор іноді виконувався на комплектах з декількох мікросхем, але зараз від такого підходу вже практично відмовилися. Мікросхема процесора обов'язково має виводи трьох шин: шини адреси, шини даних і шини управління. Іноді деякі сигнали і шини мультиплексуються, щоб зменшити кількість виводів мікросхеми процесора.

Найважливіші характеристики процесора - це кількість розрядів його шини даних, кількість розрядів його шини адреси і кількість керуючих сигналів у шині управління. Розрядність шини даних визначає швидкість роботи системи. Розрядність шини адреси визначає припустиму складність системи. Кількість ліній управління визначає розмаїтість режимів обміну й ефективність обміну процесора з іншими пристроями системи.

Крім виводів для сигналів трьох основних шин процесор завжди має вивід (чи два виводи) для підключення зовнішнього тактового сигнАЛП чи кварцового резонатора (CLK), тому що процесор завжди є тактованим пристроєм. Чим більша тактова частота процесора, тим він швидше працює, тобто тим швидше виконує команди. Утім, швидкодія процесора визначається не тільки тактовою частотою, але й особливостями його структури. Сучасні процесори виконують більшість команд за один такт і мають засоби для паралельного виконання декількох команд. Тактова частота процесора не зв'язана прямо і жорстко зі швидкістю обміну магістраллю, тому що швидкість обміну магістраллю обмежена затримками поширення сигналів і спотвореннями сигналів на магістралі. Тобто тактова частота процесора визначає тільки його внутрішню швидкодію, а не зовнішню. Іноді тактова частота процесора має нижню і верхню межі. При перевищенні верхньої межі частоти можливе перегрівання процесора, а також збої, причому, що саме неприємне, такі, що виникають не завжди і нерегулярно. Так що зі зміною цієї частоти треба бути дуже обережним.

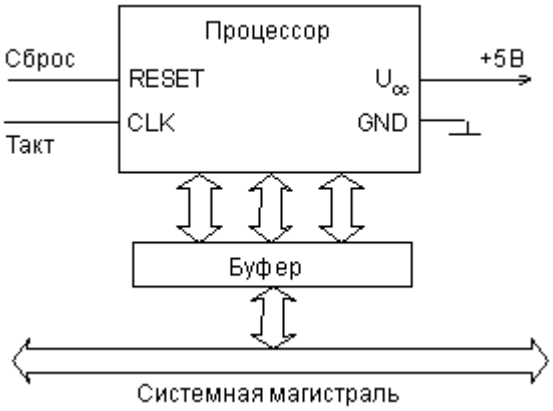


Рис. 6.1. Схема увімкнення процесора.

Ще один важливий сигнал, що є в кожному процесорі, - це сигнал початкової ініціалізації RESET. При увімкненні живлення, при аварійній

ситуації чи зависанні процесора подача цього сигнАЛП приводить до ініціалізації процесора, змушує його приступити до виконання програми початкового запуску. Аварійна ситуація може бути викликана завадами колами живлення і "землі", збоями в роботі пам'яті, зовнішніми іонізуючими випромінюваннями і ще безліччю причин. У результаті процесор може втратити контроль над виконуваною програмою і зупинитися на якійсь адресі. Для виходу з цього стану саме і використовується сигнал початкової ініціалізації. Цей же вхід може використовуватися для оповіщення процесора про те, що напруга живлення стала нижчою за встановлену межу. У такому випадку процесор переходить до виконання програми збереження важливих даних. По суті, цей вхід - це особливий різновид радіального переривання.

Іноді в мікросхеми процесора існує ще один-два входи радіальних переривань для обробки особливих ситуацій (наприклад, для переривання від зовнішнього таймера).

Шина живлення сучасного процесора звичайно має одну напругу живлення (+5В чи +3,3В) і загальний провід ("землю"). Перші процесори нерідко вимагали декількох напруг живлення. У деяких процесорах передбачений режим зниженого енергоспоживання. Узагалі, сучасні мікросхеми процесорів, особливо з високими тактовими частотами, споживають досить велику потужність. У результаті для підтримки нормальної робочої температури корпуса на них нерідко приходиться встановлювати радіатори, чи вентилятори навіть спеціальні мікрохолодильники.

Для підключення процесора до магістралі використовуються буферні мікросхеми, що забезпечують, якщо необхідно, демультиплексування сигналів і електричне буферизування сигналів магістралі. Іноді протоколи обміну системною магістраллю і шинами процесора не збігаються між собою, тоді буферні мікросхеми ще і узгоджують ці протоколи один з одним. Іноді в мікропроцесорній системі використовується кілька магістралей (системних і локальних), тоді для кожної з магістралей застосовується свій буферний вузол. Така структура характерна, наприклад, для персональних комп'ютерів.

Після увімкнення живлення процесор переходить на першу адресу програми початкового пуску і виконує цю програму. Дана програма попередньо записана в постійну (енергонезалежну) пам'ять. Після завершення програми початкового пуску процесор починає виконувати основну програму, що знаходиться в постійній чи оперативній пам'яті, для чого вибирає по черзі всі команди. Від цієї програми процесор можуть відволікати зовнішні переривання чи запити на ПДП. Команди з пам'яті процесор вибирає за допомогою циклів читання магістраллю. При необхідності процесор записує дані в пам'ять чи у пристрої вводу/виводу за допомогою циклів запису або ж читає дані з пам'яті чи з пристроїв вводу/виводу за допомогою циклів читання.

Таким чином, основні функції будь-якого процесора наступні:

- вибірка (читання) виконуваних команд;

- ввід (читання) даних з чи пам'яті пристрою вводу/виводу;
- вивід (запис) даних у чи пам'ять у пристрої вводу/виводу;
- обробка даних (операндів), у тому числі арифметичні операції над ними;
- адресація пам'яті, тобто завдання адреси пам'яті, з яким буде відбуватися обмін;
- обробка переривань і режиму прямого доступу.

Спрощено структуру мікропроцесора можна представити в наступному виді (Рис. 6.2).

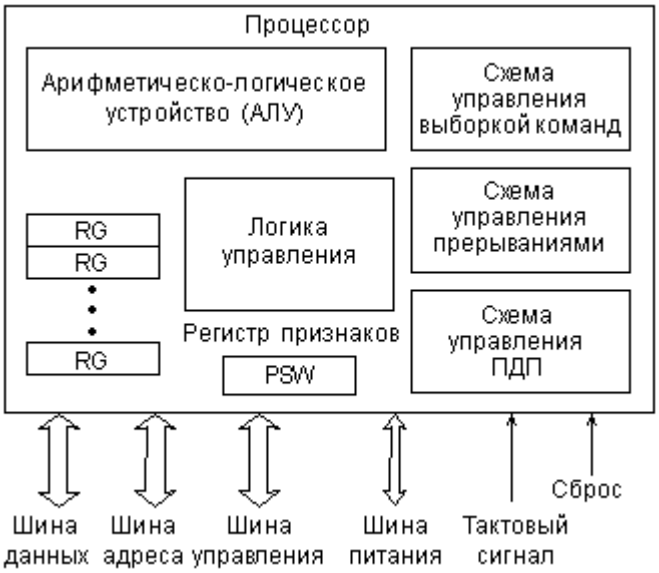


Рис. 6.2. Внутрішня структура мікропроцесора.

Основні функції показаних вузлів наступні:

Схема управління вибіркою команд виконує читання команд із пам'яті і їхню дешифрацію. У перших мікропроцесорах було неможливо одночасне виконання попередньої команди і вибірка наступної команди, тому що процесор не міг поєднувати ці операції. Але вже в 16-розрядних процесорах з'являється так званий конвеєр (черга) команд, що дозволяє вибирати кілька наступних команд, поки виконується попередня. Два процеси йдуть паралельно, що прискорює роботу процесора. Конвеєр - це невелика внутрішня пам'ять процесора, у яку при найменшій можливості (при звільненні зовнішньої шини) записується кілька команд, які слідують за тією, що виконується. Читаються ці команди процесором у тому ж порядку, що і записуються в конвеєр (це пам'ять типу FIFO, First In - First Out, перший увійшов - перший вийшов). Правда, якщо виконувана команда припускає перехід не на наступну комірку пам'яті, а на видалену (з меншою чи більшою адресою), конвеєр не допомагає, і його приходиться скидати. Але такі команди зустрічаються в програмах порівняно рідко.

Розвитком ідеї конвеєра стало використання внутрішньої кеш-пам'яті процесора, що заповнюється командами, поки процесор зайнятий виконанням попередніх команд. Чим більший об'єм кеш-пам'яті, тим менша ймовірність того, що її вміст прийдеться скинути при команді переходу.

Зрозуміло, що обробляти команди, які знаходяться у внутрішній пам'яті, процесор може набагато швидше, ніж ті, котрі розташовані в зовнішній пам'яті. У кеш-пам'яті можуть зберігатися і дані, що обробляються в даний момент, це також прискорює роботу. Для більшого прискорення вибірки команд у сучасних процесорах застосовують поєднання вибірки і дешифрації, одночасну дешифрацію декількох команд, кілька паралельних конвеєрів команд, прогнозування команд переходів і деякі інші методи.

Арифметико-логічний пристрій (АЛП чи ALU) призначений для обробки інформації відповідно до отриманого процесором командою. Прикладами обробки можуть служити логічні операції (типу логічного "І", "АБО", "Рівнозначність" і т.д.) тобто побітові операції над операндами, а також арифметичні операції (типу додавання, віднімання, множення, ділення і т.д.). Над якими кодами відбувається операція, куди поміщується її результат - визначається виконуваною командою. Якщо команда зводиться усього лише до пересилання даних без їхньої обробки, то АЛП не бере участі у її виконанні.

Швидкодія АЛП багато в чому визначає продуктивність процесора. Причому важлива не тільки частота тактового сигналу, яким тактується АЛП, але і кількість тактів, які необхідні для виконання тієї чи іншої команди. Для підвищення продуктивності виробники прагнуть довести час виконання команди до одного такту, а також забезпечити роботу АЛП на можливо більш високій частоті. Один зі шляхів рішення цієї проблеми полягає в зменшенні кількості виконуваних АЛП команд, створення процесорів зі зменшеним набором команд (так звані RISC-процесори). Інший шлях підвищення продуктивності процесора - використання декількох паралельно працюючих АЛП.

Що стосується операцій над числами з крапкою, що плаває, і інших спеціальних складних операцій, то в системах на базі перших процесорів їх реалізували послідовністю більш простих команд, спеціальними підпрограмами, однак потім були розроблені спеціальні обчислювачі - математичні сопроцесори, що заміняли основний процесор на час виконання таких команд. У сучасних мікропроцесорах математичні сопроцесори входять у структуру як складова частина.

Регістри процесора являють собою по суті осередки дуже швидкої пам'яті і служать для тимчасового збереження різних кодів: даних, адрес, службових кодів. Операції з цими кодами виконуються гранично швидко, тому, у загальному випадку, чим більше внутрішніх регістрів, тим краще. Крім того, на швидкодію процесора сильно впливає розрядність регістрів. Саме розрядність регістрів і АЛП називається внутрішньою розрядністю процесора, що може не збігатися з зовнішньою розрядністю.

Стосовно призначення внутрішніх регістрів існує два основних підходи. Першого дотримується, наприклад, компанія Intel, що кожному регістру відводить строго визначену функцію. З одного боку, це спрощує організацію процесора і зменшує час виконання команди, але з іншого боку - знижує гнучкість, а іноді і сповільнює роботу програми. Наприклад, деякі

арифметичні операції й обмін із пристроями вводу/виводу проводяться тільки через один регістр - акумулятор, у результаті чого при виконанні деяких процедур може знадобитися кілька додаткових пересилань між регістрами. Другий підхід полягає в тому, щоб усі (чи майже усі) регістри зробити рівноправними, як , наприклад, у 16-розрядних процесорах Т-11 фірми DEC. При цьому досягається висока гнучкість, але необхідне ускладнення структури процесора. Існують і проміжні рішення, зокрема, у процесорі MC68000 фірми Motorola половина регістрів використовувалася для даних, і вони були взаємозамінні, а інша половина - для адрес, і вони також взаємозамінні.

Регістр ознак (регістр стану) займає особливе місце, хоча він також є внутрішнім регістром процесора. Інформація, що міститься в ньому - це не дані, не адреса, а слово стану процесора (ССП, PSW - Processor Status Word). Кожен біт цього слова (прапор) містить інформацію про результат попередньої команди. Наприклад, є біт нульового результату, що встановлюється в тому випадку, коли результат виконання попередньої команди - нуль, і очищається в тому випадку, коли результат виконання команди відмінний від нуля. Ці біти (прапори) використовуються командами умовних переходів, наприклад, командою переходу у випадку нульового результату. У цьому ж регістрі іноді містяться прапори керування, що визначають режим виконання деяких команд.

Схема управління перериваннями обробляє запит, що надходить на процесор, переривання, визначає адресу початку програми обробки переривання (адресу вектора переривання), забезпечує перехід до цієї програми після виконання поточної команди і збереження в пам'яті (у **стеці**) поточного стану регістрів процесора. По закінченні програми обробки переривання процесор повертається до перерваної програми з відновленими з пам'яті (зі стека) значеннями внутрішніх регістрів. Докладніше про стек буде розказано в наступному розділі.

Схема управління прямим доступом до пам'яті служить для тимчасового відключення процесора від зовнішніх шин і припинення роботи процесора на час надання прямого доступу до пам'яті системи того пристрою, який його запросив.

Логіка управління організує взаємодію усіх вузлів процесора, перенаправляє дані, синхронізує роботу процесора з зовнішніми сигналами, а також реалізує процедури вводу і виводу інформації.

Таким чином, у ході роботи процесора схема вибірки команд вибирає послідовно команди з пам'яті, потім ці команди виконуються, причому в разі потреби обробки даних підключається АЛП. На входи АЛП можуть подаватися оброблювані дані з пам'яті чи з внутрішніх регістрів. В внутрішніх регістрах зберігаються також коди адрес оброблюваних даних, розташованих у пам'яті. Результат обробки в АЛП змінює стан регістра ознак і записується у внутрішній регістр чи у пам'ять (як джерело, так і приймач даних вказується в складі коду команди). При необхідності інформація може

переписуватись з пам'яті (чи з пристрою вводу/виводу) у внутрішній регістр чи із внутрішнього регістра в пам'ять (чи в пристрій вводу/виводу).

Внутрішні регістри будь-якого мікропроцесора обов'язково виконують дві службові функції:

- визначають адреси в пам'яті, де знаходиться виконувана в даний момент команда (функція *лічильника команд* чи *показника команд*);
- визначають поточну адресу стеку (функція *показника стека*).

У різних процесорах для кожної з цих функцій може виділятися один чи два внутрішніх регістри. Ці два покажчики відрізняються від інших не тільки своїм специфічним, службовим, системним призначенням, але й особливим способом зміни вмісту. Їхній уміст програми можуть змінювати тільки у випадку крайньої необхідності, тому що будь-яка помилка при цьому грозить порушенням роботи комп'ютера, зависанням і псуванням умісту пам'яті.

Уміст показника (лічильника) команд змінюється в такий спосіб: На початку роботи системи (при увімкненні живлення) у нього заноситься раз і назавжди встановлене значення. Це перша адреса програми початкового запуску. Потім після вибірки з пам'яті кожної наступної команди значення покажчика команд автоматично збільшується (інкрементується) на одиницю (чи на два в залежності від формату команд і типу процесора). Тобто наступна команда буде вибиратися з наступної по прядку адреси пам'яті. При виконанні команд переходу, що порушують послідовний перебір адрес пам'яті, у покажчик команд примусово записується нове значення - нова адреса в пам'яті, починаючи з якої адреси команд знову будуть перебиратися послідовно. Така ж зміна вмісту покажчика команд відбувається при виклику підпрограми і поверненні з неї чи при початку обробки переривання і після його закінчення.

Основна функція будь-якого процесора, заради якої він і створюється, - це виконання команд. Система команд, виконуваних процесором, являє собою щось подібне до таблиці істинності логічних елементів чи таблиці режимів роботи більш складних логічних мікросхем. Тобто вона визначає логіку роботи процесора і його реакцію на ті чи інші комбінації зовнішніх подій.

Написання програм для мікропроцесорної системи - найважливіший і часто найбільш трудомісткий етап розробки такої системи. А для створення ефективних програм необхідно мати хоча б саме загальне уявлення про систему команд використовуваного процесора. Самі компактні і швидкі програми і підпрограми створюються мовою Асемблер, використання якого без знання системи команд абсолютно неможливо, адже мова Асемблер - символьний запис цифрових кодів машинної мови, кодів команд процесора. Звичайно, для розробки програмного забезпечення існують усілякі програмні засоби. Користатися ними звичайно можна і без знання системи команд процесора. Найчастіше застосовуються мови програмування високого рівня, такі як Паскаль і Сі. Однак знання системи команд і мови Асемблер дозволяє в кілька разів підвищити ефективність деяких найбільш важливих частин

програмного забезпечення будь-якої мікропроцесорної системи - від мікроконтролера до персонального комп'ютера.

Саме тому в даному розділі ми розглянемо основні типи команд, що існують в більшості процесорів, і особливості їхнього застосування.

Кожна команда, вибрана (прочитана) з пам'яті процесором, визначає алгоритм поведінки процесора на найближчі кілька тактів. Код команди говорить про те, яку операцію має бути виконано процесором і з якими ***операндами*** (тобто кодами даних), де узяти вихідну інформацію для виконання команди і куди помістити результат (якщо необхідно). Код команди може займати від одного до декількох байт, причому процесор довідається про те, скільки байтів команди йому треба читати, з першого прочитаного їм байта чи слова. У процесорі код команди розшифровується і перетвориться в набір мікрооперацій, виконуваних окремими вузлами процесора. Але виробнику мікропроцесорних систем це знання не занадто важливе, йому важливий тільки результат виконання тієї чи іншої команди.

6.1. Адресація операндів

Велика частина команд процесора працює з кодами даних (операндами). Одні команди вимагають вхідних операндів (одного чи двох), інші видають вихідні операнди (частіше один операнд). Вхідні операнди називаються ще операндами-джерелами, а вихідні називаються операндами-приймачами. Усі ці коди операндів (вхідні і вихідні) повинні десь розташовуватися. Вони можуть знаходитися у внутрішніх регістрах процесора (найбільш зручний і швидкий варіант). Вони можуть розташовуватися в системній пам'яті (найпоширеніший варіант). Нарешті, вони можуть знаходитися в пристроях вводу/виводу (найрідший випадок). Визначення місця положення операндів відбувається кодом команди. Причому існують різні методи, за допомогою яких код команди може визначити, відкіля брати вхідний операнд і куди поміщати вихідний операнд. Ці методи називаються**методами адресації**. Ефективність обраних методів адресації багато в чому визначає ефективність роботи всього процесора в цілому.

6.1.1. Методи адресації

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

Безпосередня адресація (Рис. 6.3) припускає, що операнд (вхідний) знаходиться в пам'яті безпосередньо за кодом команди. Операнд звичайно являє собою константу, яку треба кудись переслати, до чого додати і т.д. Наприклад, команда може полягати в тому, щоб додати число 6 до вмісту якогось внутрішнього регістра процесора. Це число 6 буде розташовуватися в пам'яті, усередині програми в адресі, що слідує за кодом даної команди додавання.

Рис. 6.3. Безпосередня адресація.

Пряма (вона ж абсолютна) адресація (Рис. 6.4) припускає, що операнд (вхідний чи вихідний) знаходиться в пам'яті за адресою, код якого знаходиться усередині програми відразу ж за кодом команди. Наприклад, команда може полягати в тому, щоб очистити (зробити нульовим) уміст комірки пам'яті з адресою 1000000. Код цієї адреси 1000000 буде розташовуватися в пам'яті, усередині програми в наступній адресі за кодом даної команди очищення.

Рис. 6.4. Пряма адресація.

Регістрова адресація (Рис. 6.5) припускає, що операнд (вхідний чи вихідний) знаходиться у внутрішньому регістрі процесора. Наприклад, команда може полягати в тому, щоб переслати число з нульового регістра в перший. Номера обидвох регістрів (0 і 1) будуть визначатися кодом команди пересилання.

Непрямо-реєстрова (вона ж непряма) адресація припускає, що у внутрішньому регістрі процесора знаходиться не сам операнд, а його адреса в пам'яті (Рис. 3.4). Наприклад, команда може полягати в тому, щоб очистити комірку пам'яті з адресою, що знаходиться в нульовому регістрі. Номер цього регістра (0) буде визначатися кодом команди очищення.

Рис. 3.3. Регістрова адресація.

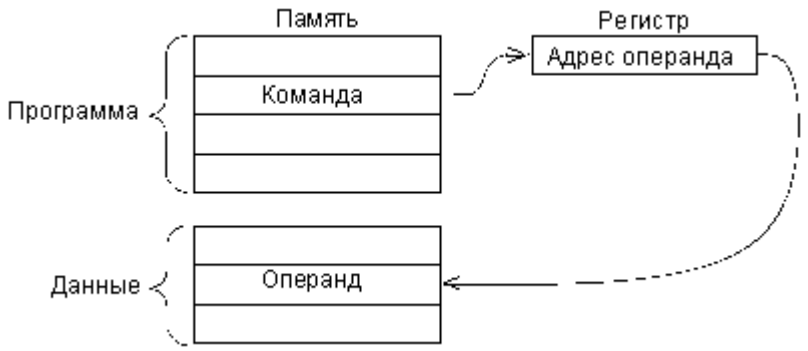


Рис. 6.5. Косвенная адресация.

Рідше зустрічаються ще два методи адресації.

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

Автодекрементна адресація працює подібно до автоінкрементної, але тільки вміст обраного регістра зменшується на одиницю чи на два *перед* виконанням команди. Ця адресація також зручна при обробці масивів даних. Спільне використання автоінкрементної та автодекрементної адресацій дозволяє організувати пам'ять стекового типу.

З інших розповсюджених методів адресації можна згадати про індексні методи, що припускають для обчислення адреси операнда додавання до вмісту регістра заданої константи (індексу). Код цієї константи розташовується в пам'яті безпосередньо за кодом команди.

Відзначимо, що вибір того чи іншого методу адресації в значній мірі визначає час виконання команди. Найшвидша адресація - це регістрова, тому що вона не вимагає додаткових циклів обміну магістраллю. Якщо ж адресація вимагає звертання до пам'яті, то час виконання команди буде збільшуватися за рахунок тривалості необхідних циклів звертання до пам'яті. Зрозуміло, що чим більше внутрішніх регістрів у процесора, тим частіше і вільніше можна застосовувати регістрову адресацію, і тем швидше буде працювати система в цілому.

6.1.2. Сегментування пам'яті.

Говорячи про адресацію, не можна обійти питання про **сегментування** пам'яті, яке застосовується в деяких процесорах, наприклад у процесорах IBM PC-сумісних персональних комп'ютерів.

У процесорі Intel 8086 сегментування пам'яті організоване в такий спосіб:

Уся пам'ять системи представляється не у виді безупинного простору, а у виді декількох кусків - сегментів заданого розміру (по 64 Кбайти), положення яких у просторі пам'яті можна змінювати програмним шляхом.

Для збереження кодів адрес пам'яті використовуються не окремі регістри, а пари регістрів:

- **сегментний регістр** визначає адресу початку сегменту (тобто місце сегмента в пам'яті);
- **регістр показника** (регістр зсуву) визначає положення робочої адреси усередині сегмента.

При цьому фізична 20-розрядна адреса пам'яті, яка виставляється на зовнішню шину адреси, утвориться так, як показано на Рис. 6.6, тобто шляхом додавання зсуву й адреси сегмента зі зсувом на 4 біти. Положення цієї адреси в пам'яті показане на Рис. 6.7.

Сегмент може починатися тільки на 16-байтній межі пам'яті (тому що адреса початку сегменту, по суті, має чотири молодших нульових розряди, як видно з Рис. 6.6), тобто з адреси, яка кратна 16. Ці припустимі межі сегментів називаються границями параграфів.

Відзначимо, що ввід сегментування, насамперед, зв'язане з тим, що внутрішні регістри процесора 16-розрядні, а фізична адреса пам'яті 20-розрядний (16-розрядна адреса дозволяє використовувати пам'ять тільки в 64 Кбайт, що явно недостатньо). У процесорі, який з'явився в той самий час, MC68000 фірми Motorola внутрішні регістри 32-розрядні, тому там проблеми сегментування пам'яті не виникає.

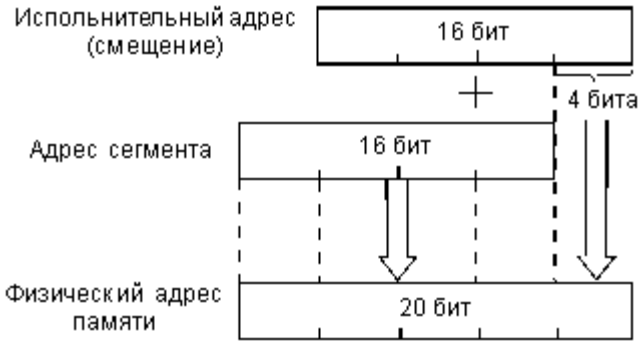


Рис. 6.6. Формування фізичної адреси пам'яті з адреси сегмента і зсуву.

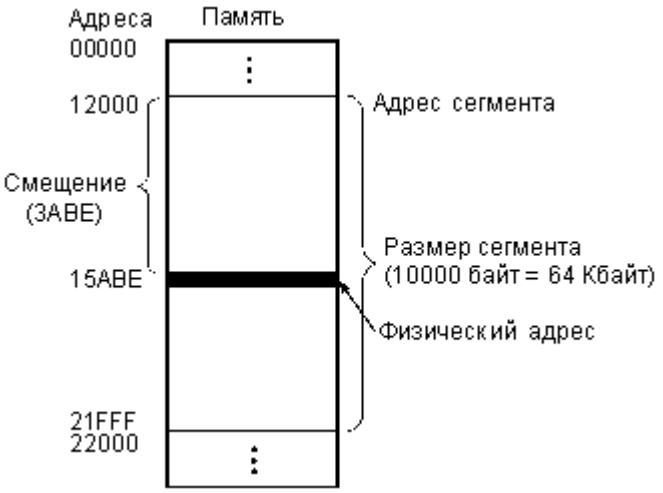


Рис. 6.7. Фізична адреса в сегменті (усі коди - шіснадцяткові).

Застосовуються і більш складні методи сегментування пам'яті. Наприклад, у процесорі Intel 80286 у так званому захищеному режимі адреса пам'яті обчислюється відповідно до Рис. 6.8.

У сегментному регістрі в даному випадку зберігається не базова (початкова) адреса сегментів, а коди селекторів, що визначають адреси в пам'яті, по яких зберігаються дескриптори (тобто описи) сегментів. Область пам'яті з дескрипторами називається таблицею дескрипторів. Кожен дескриптор сегмента містить базову адресу сегмента, розмір сегмента (від 1 до 64 Кбайт) і його атрибути. Базова адреса сегмента має розрядність 24 біт, що забезпечує адресацію 16 Мбайт фізичної пам'яті.

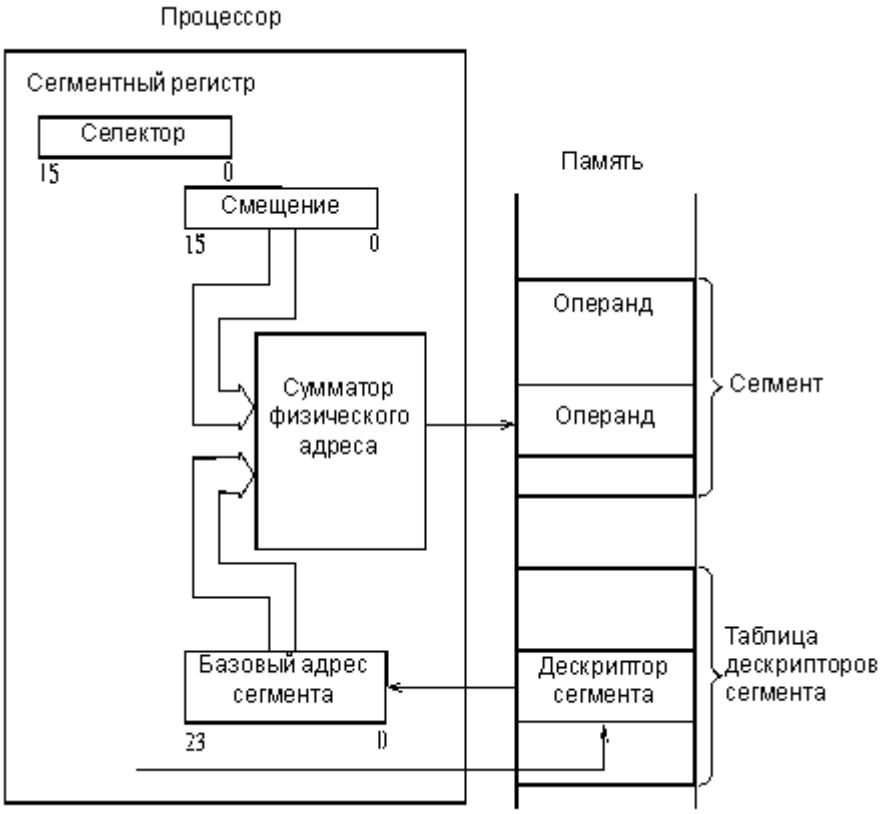


Рис. 6.8. Адресація пам'яті в захищеному режимі процесора Intel 80286.

Таким чином, на суматор, що обчислює фізичну адресу пам'яті, подається не вміст сегментного регістра, як у попередньому випадку, а базова адреса сегмента з таблиці дескрипторів.

Ще складніший метод адресації пам'яті із сегментуванням використаний у процесорі Intel 80386 і в більш пізніх моделях процесорів фірми Intel. Цей метод ілюструється Рис. 6.9.

Адреса пам'яті (фізична адреса) обчислюється в три етапи. Спочатку обчислюється так звана ефективна адреса (32-розрядна) шляхом підсумовування трьох компонентів: бази, індексу і зсуву (Base, Index, Displacement), причому можливе множення індексу на масштаб (Scale). Ці компоненти мають наступний зміст:

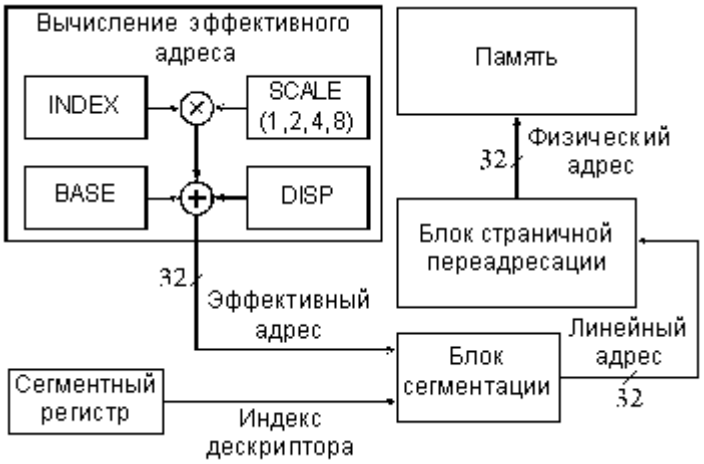


Рис. 6.9. Формування фізичної адреси пам'яті процесора 80386 у захищеному режимі.

- *зсув* - це 8-, 16- чи 32-розрядне число, включене у команду.
- *база* - це вміст базового регістра процесора. Звичайно воно використовується для вказівки на початок деякого масиву.
- *індекс* - це вміст індексного регістра процесора. Переважно воно використовується для вибору одного з елементів масиву.
- *масштаб* - це множник (він може бути рівним 1, 2, 4 чи 8), зазначений у коді команди, на який перед підсумовуванням з іншими компонентами збільшується індекс. Він використовується для вказівки розміру елемента масиву.

Потім спеціальний блок сегментації обчислює 32-розрядну *лінійну адресу*, що представляє собою суму базової адреси сегмента із сегментного регістра з ефективною адресою. Нарешті, фізична 32-бітна адреса пам'яті утвориться шляхом перетворення лінійної адреси блоком сторінкової переадресації, що здійснює переклад лінійної адреси у фізичний сторінками по 4 Кбайти.

У будь-якому випадку сегментування дозволяє виділити в пам'яті один чи кілька сегментів для даних і один чи кілька сегментів для програм. Перехід від одного сегмента до іншого зводиться усього лише до зміни вмісту сегментного регістра. Іноді це буває дуже зручно. Але для програміста працювати із сегментованою пам'яттю звичайно складніше, ніж з

безупинною, несегментованою пам'яттю, тому що приходиться стежити за межами сегментів, за їх описом, перемиканням і т.д.

3.1.3. Адресація байтів і слів

Багато процесорів, що мають розрядність 16 чи 32, здатні адресувати не тільки ціле слово в пам'яті (16-розрядне чи 32-розрядне), але й окремі байти. Кожному байту в кожнім слові при цьому приділяється своя адреса.

Так, у випадку 16-розрядних процесорів усі слова в пам'яті (16-розрядні) мають парні адреси. А байти, що входять у ці слова, можуть мати як парні адреси, так і непарні.

Наприклад, нехай 16-розрядна комірка пам'яті має адресу 23420, і в ній зберігається код 2A5E (Рис. 6.10).

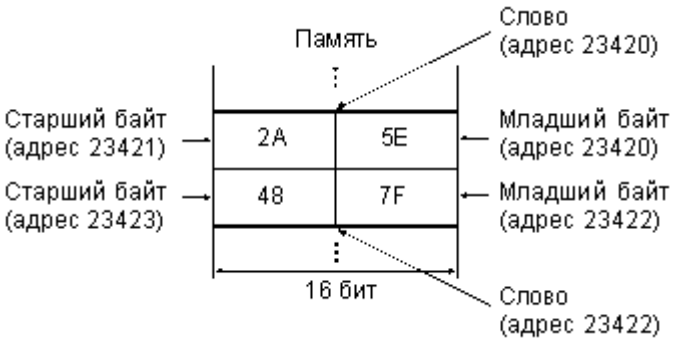


Рис. 6.10. Адресація слів і байтів.

При звертанні до цілого слова (із умістом 2A5E) процесор виставляє адресу 23420. При звертанні до молодшого байта цієї комірки (із умістом 5E) процесор виставляє ту ж саму адресу 23420, але використовує команду, що адресує байт, а не слово. При звертанні до старшого байта цієї ж комірки (із умістом 2A) процесор виставляє адресу 23421 і використовує команду, що адресує байт. Наступна 16-розрядна комірка пам'яті з вмістом 487F буде мати адресу 23422, тобто знову ж парну. Її байти будуть мати адреси 23422 і 23423.

Для можливості відрізнити байтові і слівні цикли обміну на магістралі в шині управління передбачається спеціальний сигнал байтового обміну. Для роботи з байтами в систему команд процесора вводяться спеціальні команди чи передбачаються методи байтової адресації.

6.3. Система команд процесора

У загальному випадку система команд процесора містить у собі наступні чотири основні групи команд:

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

Команди пересилки даних не вимагають виконання ніяких операцій над операндами. Операнди просто пересилаються (точніше, копіюються) із джерела (Source) у приймач (Destination). Джерелом і приймачем можуть

бути внутрішні регістри процесора, комірки пам'яті чи пристрою вводу/виводу. АЛП у даному випадку не використовується.

Арифметичні команди виконують операції додавання, віднімання, множення, ділення, збільшення на одиницю (інкементування), зменшення на одиницю (декрементування) і т.д. Цим командам потрібно один чи два вхідних операнди. Формують команди один вихідний операнд.

Логічні команди виконують над операндами логічні операції, наприклад, логічне І, логічне АБО, операцію "Нерівнозначність", очищення, інверсію, різноманітні зсуви (вправо, вліво, арифметичний зсув, циклічний зсув). Цим командам, як і арифметичним, потрібно один чи два вхідних операнди, і формують вони один вихідний операнд.

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

Відповідно до результату кожної виконаної команди встановлюються чи очищаються біти регістра стану процесора (PSW). Але треба пам'ятати, що не всі команди змінюють усі наявні в PSW прапори. Це визначається особливостями кожного конкретного процесора.

У різних процесорів системи команд істотно відрізняються, але в своїй основі вони дуже схожі. Кількість команд у процесорів також є різною. Наприклад, у згадуваного вже процесора MC68000 всього 61 команда, а в процесора 8086 - 133 команди. У сучасних потужних процесорів кількість команд досягає декількох сотень. У той же час існують процесори зі скороченим набором команд (так звані RISC-процесори), у яких за рахунок максимального скорочення кількості команд досягається збільшення ефективності і швидкості їхнього виконання.

Розглянемо тепер особливості чотирьох виділених груп команд процесора більш докладно.

6.3.1. Команди переслки даних

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

- завантаження (запис) вмісту у внутрішні регістри процесора;
- збереження в пам'яті умісту внутрішніх регістрів процесора;
- копіювання вмісту з однієї області пам'яті в іншу;
- запис у пристрої вводу/виводу і читання з пристроїв вводу/виводу.

У деяких процесорах (наприклад, Т-11) усі ці функції виконуються однією єдиною командою MOV (для байтових пересилань - MOVB) але з різними методами адресації операндів.

В інших процесорах крім команди MOV існує ще кілька команд для виконання перерахованих функцій. Наприклад, для завантаження регістрів можуть використовуватися команди завантаження, причому для різних регістрів - різні команди (їхні позначення звичайно будуються з використанням слова LOAD - завантаження). Часто виділяються спеціальні команди для збереження в стеці і для вилучення зі стека (PUSH - зберегти в стеку, POP - витягти зі стеку). Ці команди виконують пересилання з автоінкрементною і з автодекрементною адресацією (навіть якщо ці режими адресації не передбачені в процесорі в явному виді).

Іноді в систему команд уводиться спеціальна команда MOVS для рядкового (чи ланцюгового) пересилання даних (наприклад, у процесорі 8086). Ця команда пересилає не одне слово чи байт, а задану кількість слів чи байтів (MOVSB), тобто ініціює не один цикл обміну магістраллю, а декілька. При цьому адреса пам'яті, з якою відбувається взаємодія, збільшується на 1 чи на 2 після кожного звертання чи зменшується на 1 чи на 2 після кожного звертання. Тобто в неявному виді застосовується автоінкрементна чи автодекрементна адресація.

У деяких процесорах (наприклад, у процесорі 8086) спеціально виділяються функції обміну з пристроями вводу/виводу. Команда IN використовується для вводу (читання) інформації з пристрою вводу/виводу, а команда OUT використовується для виводу (запису) у пристрій уводу/виводу. Обмін інформацією в цьому випадку відбувається між регістром-акумулятором і пристроєм уводу/виводу. У більш просунутих процесорах цього ж сімейства (починаючи з процесора 80286) додано команди рядкового (ланцюгового) вводу (команда INS) і рядкового виводу (команда OUTS). Ці команди дозволяють пересилати цілий масив (рядок) даних з пам'яті в пристрій вводу/виводу (OUTS) чи з пристрою вводу/виводу в пам'ять (INS). Адреса пам'яті після кожного звертання збільшується чи зменшується (як і у випадку з командою MOVS).

Також до команд пересилання даних відносяться команди обміну інформацією (їхнє позначення будується на основі слова Exchange). Може бути передбачений обмін інформацією між внутрішніми регістрами, між двома половинами одного регістра (SWAP) чи між регістром і коміркою пам'яті.

6.3.2. Арифметичні команди

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

- команди операцій з фіксованою комою (додавання, віднімання, множення, ділення);
- команди операцій з плаваючою комою (додавання, віднімання, множення, ділення);
- команди очищення;
- команди інкременту і декременту;

- команда порівняння.

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

Команди операцій з плаваючою комою, (крапкою) використовують формат подання чисел з порядком і мантисою (звичайно ці числа займають дві послідовні комірки пам'яті). У сучасних потужних процесорах набір команд із плаваючою комою не обмежується тільки чотирма арифметичними діями, а містить і безліч інших більш складних команд, наприклад, обчислення тригонометричних функцій, логарифмічних функцій, а також складних функцій, необхідних при обробці звуку і зображення.

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

Команди інкременту (збільшення на одиницю, INC) і **декременту** (зменшення на одиницю, DEC) також бувають дуже зручні. Їх можна, в принципі, замінити командами суми з одиницею чи віднімання одиниці, але інкремент і декремент виконуються швидше, ніж додавання і віднімання. Ці команди вимагають одного вхідного операнда, що одночасно є і вихідним операндом.

Нарешті, **команда порівняння** (позначається CMP) призначена для порівняння двох вхідних операндів. По суті, вона обчислює різницю цих двох операндів, але вихідного операнда не формує, а всього лише змінює біти в регістрі стану процесора (PSW) за результатом цього віднімання. Наступна за командою порівняння команда (переважно це команда переходу) буде аналізувати біти в регістрі стану процесора і виконувати дії в залежності від їх значень (про команди переходу мова піде в розділі 3.3.4). У деяких процесорах передбачені команди ланцюгового порівняння двох послідовностей операндів, що знаходяться в пам'яті (наприклад, у процесорі 8086 і сумісних з ним).

6.3.3. Логічні команди

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

- логічне І, логічне АБО, додавання за модулем 2 (Операція "Нерівнозначність");

- логічні, арифметичні і циклічні зсуви;
- перевірка бітів і операндів;
- встановлення та очищення бітів (прапорів) регістра стану процесора (PSW).

Команди логічних операцій дозволяють побітно обчислювати основні логічні функції від двох вхідних операндів. Крім того, операція І (AND) використовується для примусового очищення заданих бітів (у якості одного з операндів при цьому використовується код маски, у якому розряди, що вимагають очищення, встановлені в нуль). Операція АБО (OR) застосовується для примусового встановлення заданих бітів (у якості одного з операндів при цьому використовується код маски, у якому розряди, що вимагають встановлення в одиницю, дорівнюють одиниці). Операція "Нерівнозначність" (XOR) використовується для інверсії заданих бітів (у якості одного з операндів при цьому застосовується код маски, у якому біти, що підлягають інверсії, встановлені в одиницю). Команди вимагають двох вхідних операндів і формують один вихідний операнд.

Команди зсувів дозволяють побітно зсувати код операнда вправо (убік молодших розрядів) чи вліво (убік старших розрядів). Тип зсуву (логічний, арифметичний чи циклічний) визначає, яке буде нове значення старшого біта (при зсуві вправо) чи молодшого біта (при зсуві вліво), а також визначає, чи буде десь збережене колишнє значення старшого біта (при зсуві вліво) чи молодшого біта (при зсуві вправо). Наприклад, при логічному зсуві вправо в старшому розряді коду операнда встановлюється нуль, а молодший розряд записується як прапор переносу в регістр стану процесора. А при арифметичному зсуві вправо значення старшого розряду зберігається незмінним (нулем чи одиницею), молодший розряд також записується як прапор переносу.

Циклічні зсуви дозволяють зсувати біти коду операнда по колу (по годинниковій стрілці при зсуві вправо чи проти годинникової стрілки при зсуві вліво). При цьому в кільце зсуву може входити або не входити прапор переносу. У біт прапора переносу (якщо він використовується) записується значення старшого біта при циклічному зсуві вліво і молодшого біта при циклічному зсуві вправо. Відповідно, значення біта прапора переносу буде переписуватися в молодший розряд при циклічному зсуві вліво й у старший розряд при циклічному зсуві вправо.

Для прикладу на Рис. 6.13 показані дії, виконувані командами зсуву вправо.

Команди перевірки бітів і операндів призначені для встановлення чи очищення бітів регістра стану процесора в залежності від значення обраних бітів чи всього операнда в цілому. Вихідного операнда команди не формують. Команда перевірки операнда (TST) перевіряє весь код операнда в цілому на рівність нулю і на знак (на значення старшого біта), вона вимагає тільки одного вхідного операнда. Команда перевірки біта (BIT) перевіряє тільки окремі біти, для вибору яких в якості другого операнда

використовується код маски. У коді маски бітам, які перевіряються, основного операнда повинні відповідати одиничні розряди.

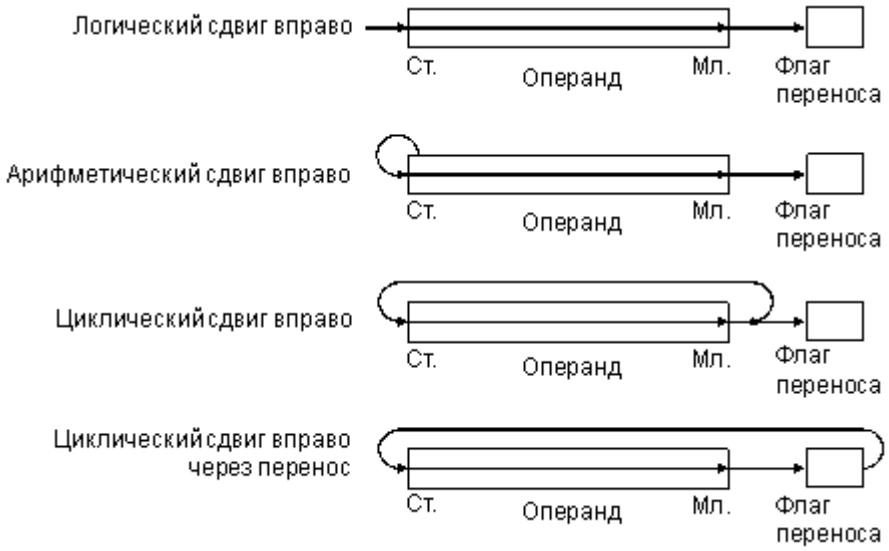


Рис. 6.13. Команди зсувів вправо.

Нарешті, команди **встановлення та очищення бітів регістра стану процесора** (тобто прапорів) дозволяють установити або очистити будь-який прапор, що буває дуже зручно. Кожному прапору звичайно відповідають дві команди, одна з яких встановлює його в одиницю, а інша скидає в нуль. Наприклад, прапору переносу C (від Carry) будуть відповідати команди CLC (очищення) і SEC чи STC (установка).

6.3.4. Команди переходів

Команди переходів призначені для організації всіляких циклів, розгалужень, викликів підпрограм і т.д., тобто вони порушують послідовний хід виконання програми. Ці команди записують у регістр-лічильник команд нове значення і тим самим викликають перехід процесора не до наступної за даною команді, а до будь-якої іншої команди в пам'яті програм. Деякі команди переходів передбачають надалі повернення назад, у місце, з якої був зроблений перехід, інші не передбачають цього. Якщо повернення передбачене, то поточні параметри процесора зберігаються в стеці. Якщо повернення не передбачене, то поточні параметри процесора не зберігаються.

Команди переходів без повернення поділяються на дві групи:

- команди безумовних переходів;
- команди умовних переходів.

У позначеннях цих команд використовуються слова Branch (розгалуження) і Jump (стрибок).

Команди безумовних переходів викликають перехід на нову адресу незалежно ні від чого. Вони можуть викликати перехід на зазначену величину зсуву (вперед чи назад) або на зазначену адресу пам'яті. Величина зсуву чи нове значення адреси вказуються як вхідного операнда.

Команди умовних переходів викликають перехід не завжди, а тільки при виконанні заданих умов. Як такі умови зазвичай виступають значення

прапорів у регістрі стану процесора (PSW). Тобто умовою переходу є результат попередньої операції, що змінює значення прапорів. Усього таких умов переходу може бути від 4 до 16. Кілька прикладів команд умовних переходів:

- перехід, якщо дорівнює нулю;
- перехід, якщо не дорівнює нулю;
- перехід, якщо є переповнення;
- перехід, якщо немає переповнення;
- перехід, якщо більше нуля;
- перехід, якщо менше чи дорівнює нулю.

Якщо умова переходу виконується, то відбувається завантаження в регістр-лічильник команд нового значення. Якщо ж умова переходу не виконується, лічильник команд просто нарощується, і процесор вибирає і виконує послідовно наступну команду.

Спеціально для перевірки умов переходу застосовується команда порівняння (CMP), яка передує команді умовного переходу (чи навіть декільком командам умовних переходів). Але прапори можуть встановлюватися і будь-якою іншою командою, наприклад командою пересилання даних, будь-якою арифметичною чи логічною командою. Відзначимо, що самі команди переходів прапори не змінюють, що саме і дозволяє ставити кілька команд переходів одну за іншою.

Спільне використання декількох команд умовних і безумовних переходів дозволяє процесору виконувати розгалужені алгоритми будь-якої складності. Для прикладу на Рис. 6.14 показано розгалуження програми на двох гілок з наступною сполученням, а на Рис. 6.15 - розгалуження на три гілки з наступною сполученням.

Команди переходів з подальшим поверненням у місце, з якого був виконаний перехід, застосовуються для виконання підпрограм, тобто допоміжних програм. Ці команди називаються також командами виклику підпрограм (розповсюджена назва - CALL). Використання підпрограм дозволяє спростити структуру основної програми, зробити її більш логічною, гнучкою, легкою для написання і налагодження. У той же час треба враховувати те, що широке використання підпрограм, як правило, збільшує час виконання програми.

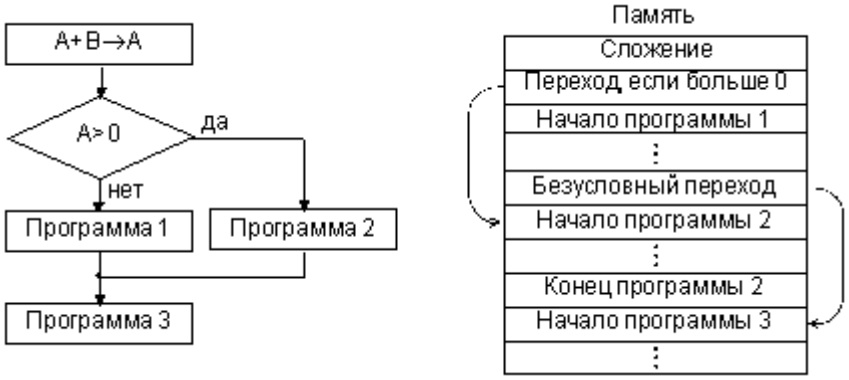


Рис. 6.14. Реалізація розгалуження на дві гілки.

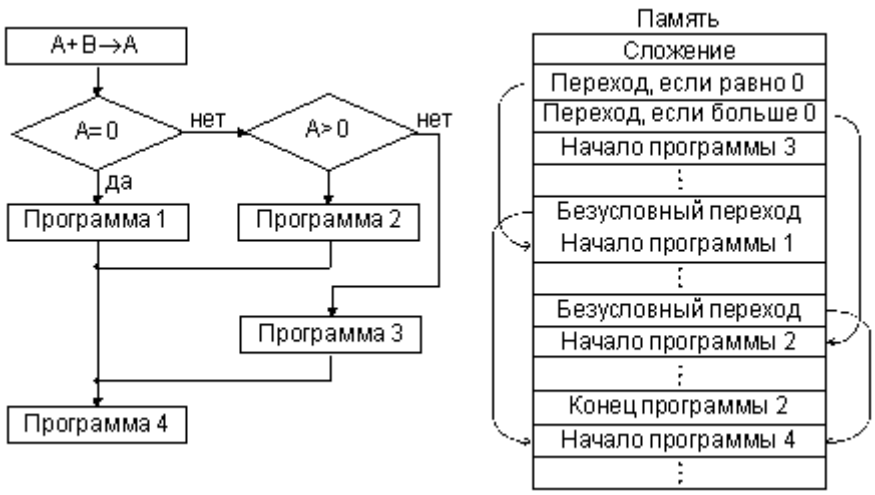


Рис. 6.15. Реалізація розгалуження на три гілки

Усі команди переходів з поверненням припускають безумовний перехід (вони не перевіряють ніяких прапорів). При цьому вони вимагають одного вхідного операнда, що може вказувати як абсолютне значення нової адреси, так і зсув, що складається з поточним значенням адреси. Поточне значення лічильника команд (поточна адреса) зберігається перед виконанням переходу в стеці.

Для зворотного повернення в місце виклику підпрограми (місце переходу) використовується спеціальна команда повернення (RET чи RTS). Ця команда витягає зі стека значення адреси команди переходу і записує його в регістр-лічильник команд.

Особливе місце серед команд переходу з поверненням займають команди переривань (розповсюджена назва - INT). Ці команди як вхідного операнда вимагають номер переривання (адресу вектора). Обслуговування таких переходів здійснюється точно так само, як і апаратних переривань. Тобто для виконання даного переходу процесор звертається до таблиці векторів переривань і одержує з неї за номером переривання адресу пам'яті, куди йому необхідно перейти. Адреса виклику переривання і вміст регістра стану процесора (PSW) зберігаються в стеці. Збереження PSW - важлива відмінність команд переривання від команд переходів з поверненням.

Команди переривань у багатьох випадках виявляються зручнішими, ніж звичайні команди переходів з поверненням. Сформувати таблицю векторів переривань можна один раз, а потім уже звертатися до неї в міру необхідності. Номер переривання відповідає номерові підпрограми, тобто номеру функції, виконуваної підпрограмою. Тому команди переривання набагато частіше включаються в системи команд процесорів, ніж звичайні команди переходів з поверненням.

Для повернення з підпрограми, викликаною командою переривання, використовується команда повернення з переривання (IRET чи RTI). Ця команда витягає зі стека збережене там значення лічильника команд і регістра стану процесора (PSW).

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

Звичайно, у даному розділі ми розглянули тільки основні команди, що найчастіше зустрічаються в процесорах. У конкретних процесорів може бути і багато інших команд, що не відносяться до перерахованих груп команд. Але вивчати їх треба вже після того, як обраний тип процесора, який підходить для задачі, що розв'язуватиметься даною мікропроцесорною системою.

ТЕМА 7. ПАМ’ЯТЬ МІКРОПРОЦЕСОРНИХ СИСТЕМ.

7.1. Функції пам'яті

Пам'ять мікропроцесорної системи виконує функцію тимчасового чи постійного збереження даних і команд. Об'єм пам'яті визначає припустиму складність виконуваних системою алгоритмів, а також до деякої міри і швидкість роботи системи в цілому. Модулі пам'яті виконуються на мікросхемах пам'яті (оперативної чи постійної). Усе частіше в складі мікропроцесорних систем використовується флеш-пам'ять (англ. - flash memory), що є енергонезалежною пам'яттю з можливістю багаторазового перезапису її вмісту.

Інформація в пам'яті зберігається в комірках, кількість розрядів яких дорівнює кількості розрядів шини даних процесора. Зазвичай вона кратна восьми (наприклад, 8, 16, 32, 64). Допустима кількість комірок пам'яті визначається кількістю розрядів шини адреси як 2^N, де N - кількість розрядів шини адреси. Найчастіше об'єм пам'яті виміряється в байтах незалежно від розрядності комірки пам'яті. Використовуються також наступні більші одиниці об'єму пам'яті: кілобайт - 2^{10} чи 1024 байта (позначається Кбайт), мегабайт - 2^{20} чи 1 048 576 байт (позначається Мбайт), гігабайт - 2^{30} байт (позначається Гбайт), терабайт - 2^{40} (позначається Тбайт) Наприклад, якщо пам'ять має 65 536 комірок, кожна з яких 16-розрядна, то говорять, що пам'ять має об'єм 128 Кбайт. Сукупність комірок пам'яті називається зазвичай *простором пам'яті* системи.

Для під'єднання модуля пам'яті до системної магістралі використовуються блоки узгодження, що містять у собі дешифратор (**селектор) адреси**, схему обробки керуючих сигналів магістралі і **буфери даних** (Рис. 7.1).

Оперативна пам'ять спілкується із системною магістраллю в циклах читання і запису, постійна пам'ять - тільки в циклах читання. Переважно в складі системи існує кілька модулів пам'яті, кожний з який працює у своїй області простору пам'яті. Селектор адреси саме і визначає, яка область адреси простору пам'яті відведена даному модулю пам'яті. Схема управління виробляє в потрібні моменти сигнали дозволу роботи пам'яті (CS) і сигнали дозволу запису в пам'ять (WR). Буфери даних передають дані від пам'яті до магістралі чи від магістралі до пам'яті.

У об'ємі пам'яті мікропроцесорної системи зазвичай виділяються кілька особливих областей, що виконують спеціальні функції.

Пам'ять програми початкового запуску завжди виконується на **ПЗП** чи флеш-пам'яті. Саме з цієї області процесор починає роботу після увімкнення живлення і після скидання його за допомогою сигналу "RESET".

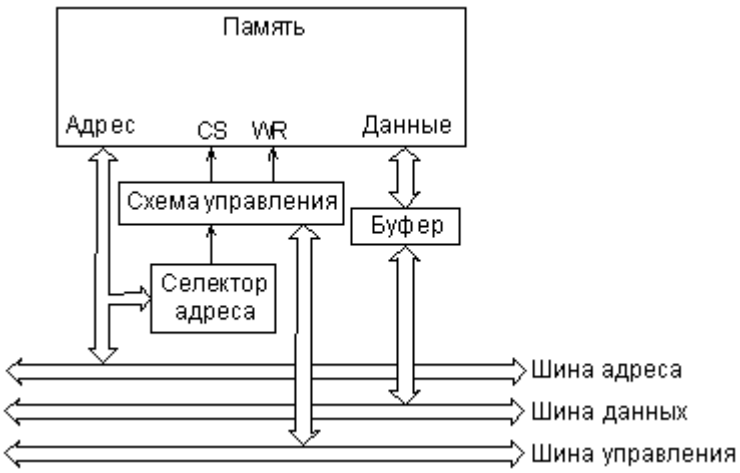


Рис. 7.1. Структура модуля пам'яті.

Пам'ять для стеку чи стек (Stack) - це частина оперативної пам'яті, яка призначена для тимчасового збереження даних у режимі LIFO (Last In - First Out).

Особливість стека в порівнянні з іншою оперативною пам'яттю - це заданий і незмінний спосіб адресації. При записі будь-якого числа (коду) у стек число записується за адресою, що визначається як уміст регістра показника стека, попередньо зменшене (декременоване) на одиницю (чи на два, якщо 16-розрядні слова розташовані в пам'яті з парними адресами). При читанні зі стека число читається з адреси, яка визначається вмістом показника стека, після чого цей уміст показника стека збільшується (інкрементується) на одиницю (чи на два). У результаті виходить, що число, записане останнім, буде прочитане першим, а число, записане першим, буде прочитане останнім. Така пам'ять називається LIFO чи пам'яттю магазинного типу (наприклад, у магазині автомата патрон, встановлений останнім, буде витягнутий першим).

Принцип дії стека показаний на Рис. 7.2 (адреси комірок пам'яті обрані умовно).

Нехай, наприклад, поточний стан показника стека 1000008, і в нього треба записати два числа (слова). Перше слово буде записане за адресою 1000006 (перед записом показник стека зменшиться на два). Друге - за адресою 1000004. Після запису вміст показника стека - 1000004. Якщо потім прочитати зі стека два слова, то першим буде прочитане слово з адреси 1000004, а після читання показник стека стане рівним 1000006. Другим буде прочитане слово з адреси 1000006, а показник стека стане рівним 1000008. Усе повернулося до вихідного стану. Перше записане слово читається другим, а друге - першим.

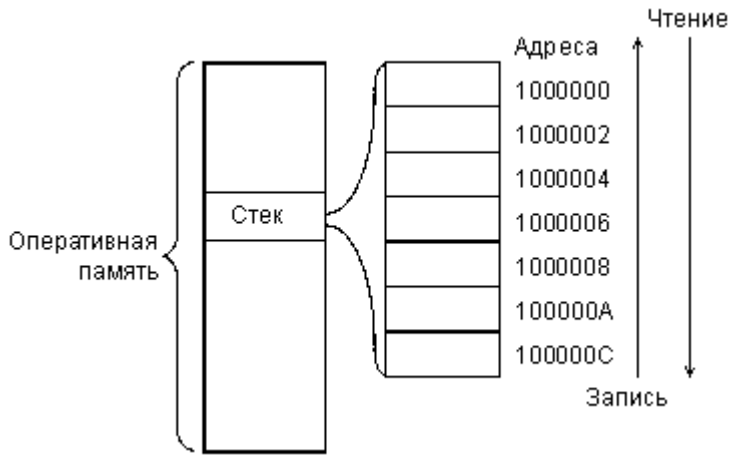


Рис. 7.2. Принцип роботи стека.

Необхідність такої адресації стає очевидною у випадку багаторазово вкладених підпрограм. Нехай, наприклад, виконується основна програма, і з неї викликається підпрограма 1. Якщо нам треба зберегти значення даних і внутрішніх регістрів основної програми на час виконання підпрограми, ми перед викликом підпрограми збережемо їх у стекові (запишемо в стек), а після її закінчення витягнемо (прочитаємо) їх зі стеку. Якщо ж з підпрограми 1 викликається підпрограма 2, то ту ж саму операцію ми проробимо з даними і умістом внутрішніх регістрів підпрограми 1. Зрозуміло, що усередині підпрограми 2 крайніми в стеці (читаються в першу чергу) будуть дані з підпрограми 1, а дані з основної програми будуть глибше. При цьому у випадку читання зі стека автоматично буде дотримуватися потрібний порядок інформації, яка читається. Те ж саме буде й у випадку, коли таких рівнів вкладення підпрограм набагато більше. Тобто те, що треба зберігати довше, ховається глибше, а те, що незабаром може знадобитися - скраю.

У системі команд будь-якого процесора для обміну інформацією зі стеком передбачені спеціальні команди запису в стек (PUSH) і читання зі стека (POP). У стеку можна містити не тільки уміст усіх внутрішніх регістрів процесора, але і вміст регістра ознак (слово стану процесора, PSW). Це дозволяє, наприклад, при поверненні з підпрограми контролювати результат останньої команди, виконаної безпосередньо перед викликом цієї підпрограми. Можна також зберігати в стеку і дані, для того щоб зручніше було передавати їх між програмами і підпрограмами. У загальному випадку, чим більша область пам'яті, відведена під стек, тим більше волі в програміста і тим складніші програми можуть виконуватися.

Наступна спеціальна область пам'яті - це **таблиця векторів переривань.**

Узагалі, поняття переривання досить багатозначне. Під перериванням у загальному випадку розуміється не тільки обслуговування запиту зовнішнього пристрою, але і будь-яке порушення послідовної роботи процесора. Наприклад, може бути передбачене переривання за фактом некоректного виконання арифметичної операції типу ділення на нуль. Або переривання може бути програмним, коли в програмі використовується команда переходу на якусь підпрограму, з якої потім піде повернення в

основну програму. В останньому випадку загальне з основним перериванням є тільки те, як здійснюється перехід на підпрограму і повернення з неї.

Будь-яке переривання обробляється через таблицю векторів (покажчиків) переривань. У цій таблиці в найпростішому випадку знаходяться адреси початку програм обробки переривань, що і називаються векторами. Довжина таблиці може бути досить великою (до кількох сотень елементів). Звичайно таблиця векторів переривань розташовується на початку простору пам'яті (у комірках пам'яті з малими адресами). Адреса кожного вектора (чи адреса початкового елемента кожного вектора) і є номером переривання.

У випадку апаратних переривань номер переривання або задається пристроєм, що запросив переривання (при векторних перериваннях), або ж задається номером лінії запиту переривань (при радіальних перериваннях). Процесор, одержавши апаратне переривання, закінчує виконання поточної команди і звертається до пам'яті в область таблиці векторів переривань, у той її рядок, що визначається номером запитаного переривання. Потім процесор читає вміст цього рядка (код вектора переривання) і переходить на адресу пам'яті, що задається цим вектором. Починаючи з цієї адреси в пам'яті повинна розташовуватися програма обробки переривання з даним номером. Наприкінці програми обробки переривань обов'язково повинна розташовуватися команда виходу з переривання, виконавши яку, процесор повертається до виконання перерваної основної програми. Параметри процесора на час виконання програми обробки переривання зберігаються в стеку.

Нехай, наприклад, процесор (Рис.7.3) виконував основну програму і команду, що знаходиться в пам'яті за адресою 5000 (умовно). У цей момент він одержав запит переривання з номером (адресою вектора) 4. Процесор закінчує виконання команди з адреси 5000. Потім він зберігає в стеці поточне значення лічильника команд (5001) і поточне значення PSW. Після цього цей процесор читає з адреси 4 пам'яті код вектора переривання. Нехай цей код дорівнює 6000. Процесор переходить на адресу пам'яті 6000 і приступає до виконання програми обробки переривання, що починається з цієї адреси. Нехай ця програма закінчується за адресою 6100. Дійшовши до цієї адреси, процесор повертається до виконання перерваної програми. Для цього він витягає зі стека значення адреси (5001), на якому його перервали, і колишнє в той момент PSW. Потім процесор читає команду з адреси 5001 і далі послідовно виконує команди основної програми.

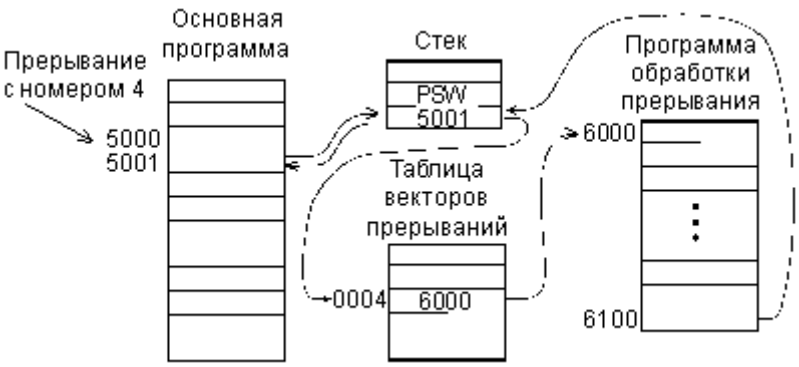


Рис. 7.3. Спрощений алгоритм обробки переривання.

Переривання у випадку аварійної ситуації обробляється точно так само, тільки адреса вектора переривання (номер рядка в таблиці векторів) жорстко прив'язаний до даного типу аварійної ситуації.

Програмне переривання теж обслуговується через таблицю векторів переривань, але номер переривання вказується в складі команди, яка викликає переривання.

Така складна, на перший погляд, організація переривань дозволяє програмісту легко змінювати програми обробки переривань, розташовувати їх у будь-якій області пам'яті, робити їх будь-якого розміру і будь-якої складності.

Під час виконання програми обробки переривання може надійти новий запит на переривання. У цьому випадку він обробляється точно так само, як описано, але основною програмою вважається перервана програма обробки попереднього переривання. Це називається багаторазовим вкладенням переривань. Механізм стека дозволяє без проблем обслуговувати це багаторазове вкладення, тому що першим зі стека витягається той код, що був збережений останнім, тобто повернення з обробки даного переривання відбувається в програму обробки попереднього переривання.

Відзначимо, що в більш складних випадках у таблиці векторів переривань можуть знаходитися не адреси початку програм обробки переривань, а так звані дескриптори (описи) переривань. Але кінцевим результатом обробки цього дескриптора все одно буде адреса початку програми обробки переривань.

Нарешті, ще одна спеціальна область пам'яті мікропроцесорної системи - це **пам'ять пристроїв, підключених до системної шини**. Таке рішення зустрічається нечасто, але іноді воно дуже зручне. Тобто процесор одержує можливість звертатися до внутрішньої пам'яті пристроїв вводу/виводу чи якихось інших під'єднаних до системної шини пристроїв, як до своєї власної системної пам'яті. Зазвичай вікно в просторі пам'яті, що виділяється для цього, не надто велике.

Всі інші частини простору пам'яті, як правило, мають універсальне призначення. У них можуть розташовуватися як дані, так і програми (зазвичай, у випадку одношинної архітектури). Іноді цей простір пам'яті використовується як єдине ціле, без усяких меж. А іноді простір пам'яті

поділяється на сегменти з програмно змінюваною адресою початку сегмента і з визначеним розміром сегмента. Обидва підходи мають свої плюси і мінуси. Наприклад, використання сегментів дозволяє захистити область програм чи даних, але зате межі сегментів можуть утрудняти розміщення великих програм і масивів даних.

На закінчення зупинимося на проблемі поділу адрес пам'яті й адрес пристроїв вводу/виводу. Існує два основних підходи до рішення цієї проблеми:

- виділення в загальному адресному просторі системи спеціальної області адрес для пристроїв вводу/виводу;
- повне роділення адресних просторів пам'яті і пристроїв вводу/виводу.

Перший підхід добрий тим, що при звертанні до пристроїв вводу/виводу процесор може використовувати ті ж команди, що служать для взаємодії з пам'яттю. Але адресний простір пам'яті повинен бути зменшений на величину адресного простору пристроїв вводу/виводу. Наприклад, при 16-розрядній шині адреси усього може бути 64К адрес. З них 56К адрес приділяється під адресний простір пам'яті, а 8К адрес - під адресний простір пристроїв вводу/виводу.

Перевага другого підходу полягає в тому, що пам'ять займає весь адресний простір мікропроцесорної системи. Для спілкування з пристроями вводу/виводу застосовуються спеціальні команди і спеціальні строби обміну на магістралі. Саме так зроблено, наприклад, у персональних комп'ютерах. Але можливості взаємодії з пристроями вводу/виводу в даному випадку істотно обмежені в порівнянні з можливостями спілкування з пам'яттю.

7.2. Організація пам'яті програм і стека

Лічильник команд у МК PIC16F8X має ширину 13 біт і здатний адресувати 8Кх14біт об'єму програмної пам'яті. Однак фізично на кристалах PIC16F83 і PIC16CR83 існує тільки 512х14 пам'яті (адреси 0000h-01FFh), а в МК PIC16F84 і PIC16CR84 - 1Кх14 пам'яті (адреси 0000h-03FFh). Звертання до адрес вище 1FFh (3FFh) фактично є адресація в тих же перші 512 адрес (перші 1К адрес).

Організація пам'яті програм і стека наведена на Рис. 7.4.

У пам'яті програм є виділені адреси. Вектор ініціалізації знаходиться за адресою 0000h, вектор переривання - за адресою 0004h. Звичайно за адресою 0004h розташовується підпрограма ідентифікації й обробки переривань, а за адресою 0000h - команда переходу на мітку, розташовану за підпрограмою обробки переривань.

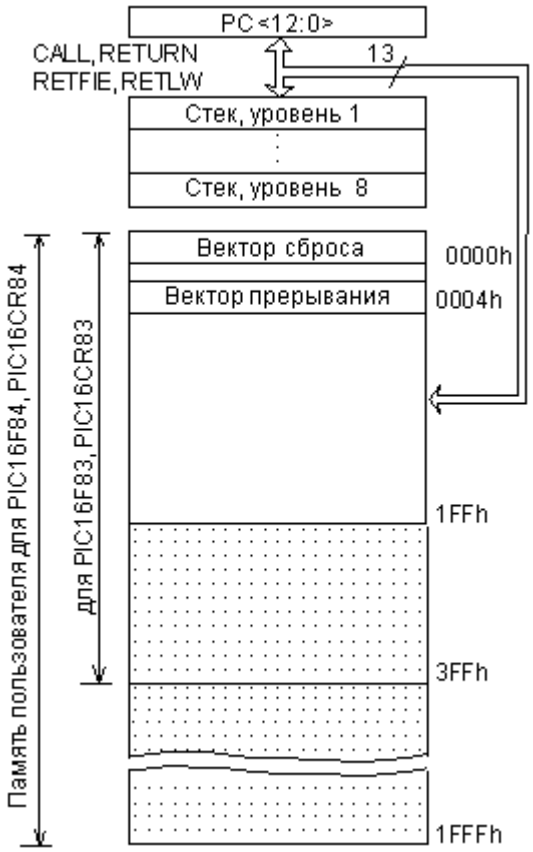


Рис. 7.4. Організація пам'яті програм і стека.

7.3. Організація пам'яті даних

Пам'ять даних МК розбита на дві області. Перші 12 адрес - це область регістрів спеціальних функцій (SFR), а друга - область регістрів загального призначення (GPR). Область SFR керує роботою приладу.

Обидві області розбиті у свою чергу на банки 0 і 1. Банк 0 вибирається шляхом обнулення біта RP0 регістра статусу (STATUS). Встановлення біта RP0 в одиницю вибирає банк 1. Кожен банк має довжину 128 байт. Однак для PIC16F83 і PIC16CR83 пам'ять даних існує тільки до адреси 02Fh, а для PIC16F84 і PIC16CR84 - до адреси 04Fh.

На Рис. 7.5 зображено організацію пам'яті даних.

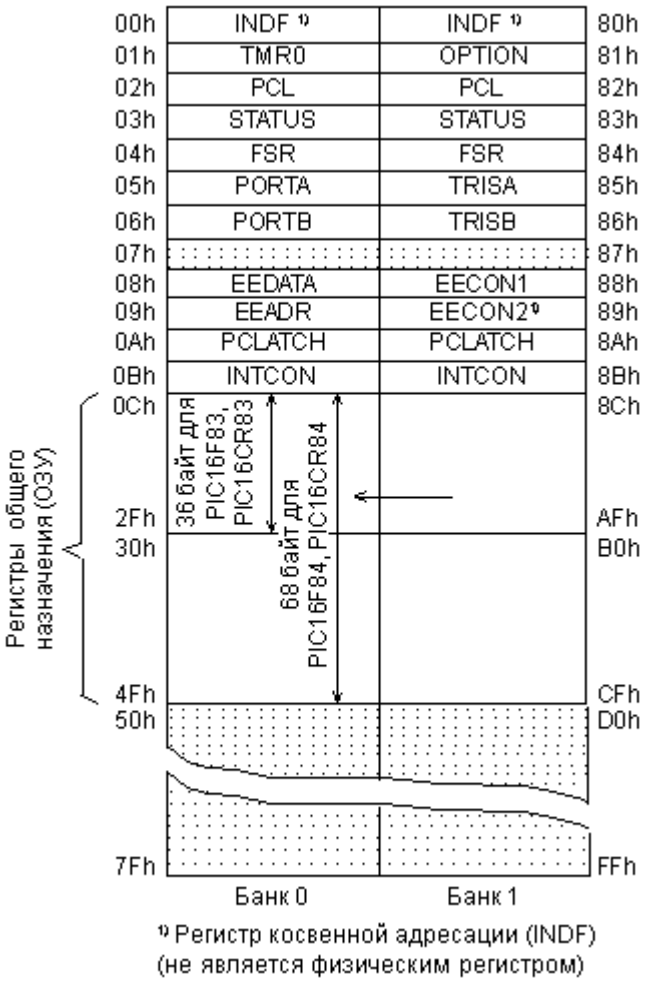


Рис. 7.5. Організація пам'яті даних.

Деякі регістри спеціального призначення продубльовані в обидвох банках, а деякі розташовані в банку 1 окремо.

Регістри з адресами 0Ch-4Fh можуть використовуватися як регістри загального призначення, які являють собою статичний ОЗП. Адреси регістрів загального призначення банку 1 відображаються на банк 0. Отже, коли встановлений банк 1, то звертання до адрес 8Ch-CFh фактично адресує банк 0.

У регістрі статусу крім біта RP0 є ще біт RB1, що дозволяє звертатися до чотирьох сторінок (банків) майбутніх модифікацій цього кристала.

До комірок ОЗП можна адресуватися прямо, використовуючи абсолютну адресу кожного регістра, або побічно, через регістр покажчик FSR. Непряма адресація використовує поточне значення розрядів RP1:RP0 для доступу до банок. Це відноситься і до EEPROM пам'яті даних. В обидвох випадках можна адресувати до 512 регістрів.

7.4. Регістри спеціального призначення

Регістр статусу (STATUS) містить ознаки операції (арифметичні прапори) АЛП, стан контролера при ініціалізації та біти вибору сторінок для пам'яті даних. Призначення бітів регістра наведене у табл. 7.1.

Табл. 7.1. Призначення бітів регістра STATUS (адреса 03h, 83h).							
R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	/TO	/PD	Z	DC	C
Біт 7	Біт 6	Біт 5	Біт 4	Біт 3	Біт 2	Біт 1	Біт 0
Біт 7: IRP: біт вибору сторінки банку даних (використовується при непрямій адресації) 0 = банк 0,1 (00h - FFh) 1 = банк 2,3 (100h - 1FFh) Біт IRP не використовується в МК підгрупи PIC16F8X							
Біти 6-5: RP1:RP0: біти вибору сторінки банку даних (використовуються при прямої адресації) 00 = банк 0 (00h - 7Fh) 01 = банк 1 (80h - FFh) 10 = банк 2 (100h - 17Fh) 11 = банк 3 (180h - 1FFh) У МК підгрупи PIC16F8X використовується тільки біт RP0							
Біт 4: /TO: біт спрацьовування сторожового таймера 1 = після увімкнення живлення, а також командами CLRWDT і SLEEP 0 = при завершенні витримки сторожового таймера							
Біт 3: /PD: біт зниження споживаної потужності 1 = після увімкнення живлення, а також командою CLRWDT 0 = за командою SLEEP							
Біт 2: Z: біт нульового результату 1 = результат арифметичної або логічної операції нульовий 0 = результат арифметичної або логічної операції ненульовий							
Біт 1: DC: біт десяткового переносу/займу (для команд ADDWF і ADDLW) 1 = має місце перенос з 4-го розряду 0 = немає переносу з 4-го розряду							
Біт 0: C: біт переносу/заема (для команд ADDWF і ADDLW) 1 = має місце перенос із самого старшого розряду 0 = немає переносу із самого старшого розряду Примітка: обчислення здійснюється шляхом додавання додаткового коду другого операнда. При виконанні команд зсуву цей біт завантажується з молодшого або старшого розряду джерела, що зсувається.							

Тут і далі: R - біт, що читається; W - біт, що записується; S - біт, що встановлюється; U - біт, що не використовується (читається як "0"); -n = 0 або 1 - значення біта після ініціалізації.

Регістр статусу доступний для будь-якої команди так само, як будь-який інший регістр. Однак якщо регістр STATUS є регістром призначення для команди, що впливає на біти Z, DC або C, то запис у ці три біти забороняється. Крім того, біти /TO і /PD встановлюються апаратно і не можуть бути записані в статус програмно. Це варто мати на увазі при

виконанні команди з використанням регістра статусу. Наприклад, команда CLRF STATUS обнулить усі біти, крім бітів /TO і /PD, а потім встановить біт Z=1. Після виконання цієї команди регістр статусу може і не мати нульового значення (через біти /TO і /PD) STATUS=000uu1uu, де u - незмінний стан. Тому рекомендується для зміни регістра статусу використовувати тільки команди бітової установки BCF, BSF, MOVWF, які не змінюють інші біти статусу. Вплив усіх команд на біти статусу розглядається в розділі "Опис системи команд".

Регістр конфігурації (OPTION) є доступним по читанню і запису регістром, містить керуючі біти для конфігурації попереднього дільника (предільника), зовнішніх переривань, таймера, а також резисторів "pull-up" на виводах PORTB. Призначення бітів регістра наведений у табл. 7.2.

Табл. 7.2. Призначення бітів регістра OPTION (адреса 81h).

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
/RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
Біт 7	Біт 6	Біт 5	Біт 4	Біт 3	Біт 2	Біт 1	Біт 0

Біт 7: /RBPU: біт установки резисторів "pull-up" на виводах PORTB
0 = резистори "pull-up" підключені
1 = резистори "pull-up" відключені

Біт 6: INTEDG: біт вибору переходу сигналу переривання
0 = переривання по спаду сигналу на виводі RB0/INT
1 = переривання по фронту сигналу на виводі RB0/INT

Біт 5: T0CS: біт вибору джерела сигналу таймера TMR0
0 = внутрішній тактовий сигнал (CLKOUT)
1 = перехід на виводі RA4/T0CKI

Біт 4: T0SE: біт вибору переходу джерела сигналу для TMR0
0 = збільшення по фронту сигналу на виводі RA4/T0CKI
1 = збільшення по спаду сигналу на виводі RA4/T0CKI

Біт 3: PSA: біт призначення предільника
0 = предільник підключений до TMR0
1 = предільник підключений до сторожового таймера WDT

Біти 2-0: PS2:PS0: біти вибору коефіцієнта ділення предільника

Значення біт	Швидкість TMR0	Швидкість WDT
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

У тому випадку, коли предільник обслуговує сторожовий таймер WDT, таймеру TMR0 призначається коефіцієнт попереднього ділення 1:1.

Регістр умов переривання (INTCON) є доступним по читанню і запису регістром, містить біти доступу для всіх джерел переривань. Призначення бітів регістра наведений у табл. 7.3.

Табл. 7.3. Призначення бітів регістра INTCON (адреси 0Bh, 8Bh).

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
Біт 7	Біт 6	Біт 5	Біт 4	Біт 3	Біт 2	Біт 1	Біт 0
Біт 7: GIE: біт дозвіл всіх переривань 0 = заборонені всі переривання 1 = дозволені всі незамасковані переривання							
Біт 6: EEIE: біт дозвіл переривання запису в EEPROM 0 = заборонені переривання запису в EEPROM 1 = дозволені переривання запису в EEPROM							
Біт 5: T0IE: біт дозвіл переривання по переповненню TMR0 0 = заборонені переривання від TMR0 1 = дозволені переривання від TMR0							
Біт 4: INTE: біт дозвіл переривань з входу RB0/INT 0 = заборонені переривання з входу RB0/INT 1 = дозволені переривання з входу RB0/INT							
Біт 3: RBIE: біт дозвіл переривань за зміною PORTB 0 = заборонені переривання за зміною PORTB 1 = дозволені переривання за зміною PORTB							
Біт 2: T0IF: біт запиту переривання за переповненням TMR0 0 = переривання за переповненням TMR0 відсутнє 1 = переривання за переповненням TMR0 має місце							
Біт 1: INTF: біт запиту переривання з входу RB0/INT 0 = переривання з входу RB0/INT відсутнє 1 = переривання з входу RB0/INT має місце							
Біт 0: RBTF: біт запиту переривання за зміною PORTB 0 = на жодному з входів RB7:RB4 стан не змінився 1 = хоча б на одному з входів RB7:RB4 змінився стан							

Біт дозволу всіх переривань GIE скидається автоматично при наступних обставинах:

- при увімкненні живлення;
- за зовнішнім сигналом /MCLR при нормальній роботі;
- за зовнішнім сигналом /MCLR у режимі SLEEP;
- при завершенні затримки таймера WDT при нормальній роботі;
- при завершенні затримки таймера WDT у режимі SLEEP.

Переривання INT може вивести процесор з режиму SLEEP, якщо перед входом у цей режим біт INTE був встановлений в одиницю. Стан біта GIE також визначає: чи буде процесор переходити на підпрограму переривання після виходу з режиму SLEEP.

Скидання бітів - запитів переривань - повинне здійснюватися відповідною програмою обробки.

ТЕМА 8. СИСТЕМА ВВОДУ-ВИВОДУ. ТАЙМЕР.

Пристрої вводу/виводу обмінюються інформацією з магістраллю за тими ж принципами, що і пам'ять. Найбільш істотна відмінність з погляду організації обміну полягає в тому, що модуль пам'яті має в адресному просторі системи багато адрес (до декількох десятків мільйонів), а пристрій вводу/виводу переважно має небагато адрес (зазвичай до десяти), а іноді і всього одну адресу.

Але модулі пам'яті системи обмінюються інформацією тільки з магістраллю, із процесором, а пристрої вводу/виводу взаємодіють ще і з зовнішніми пристроями, цифровими чи аналоговими. Тому розмаїтість пристроїв вводу/виводу незмірно більша, ніж модулів пам'яті. Часто використовуються ще й інші назви для пристроїв вводу/виводу: пристрої узгодження, контролери, карти розширення, інтерфейсні модулі та ін.

Поєднують усі пристрої вводу/виводу загальні принципи обміну з магістраллю і, відповідно, загальні принципи організації вузлів, що здійснюють узгодження з магістраллю. Спрощена структура пристрою вводу/виводу (точніше, його інтерфейсної частини) наведена на Рис. 8.1. Як і у випадку модуля пам'яті, вона обов'язково містить схему селектора адреси, схему управління для обробки стробів обміну і буфери даних.

Самі найпростіші пристрої вводу/виводу видають на зовнішній пристрій код даних у паралельному форматі і приймають із зовнішнього пристрою код даних у паралельному форматі. Такі пристрої вводу/виводу часто називають паралельними портами вводу/виводу. Вони найбільш універсальні, тобто задовольняють потреби узгодження з великою кількістю зовнішніх пристроїв, тому їх часто уводять до складу мікропроцесорної системи як стандартні пристрої. Паралельні порти зазвичай є в складі мікроконтролерів. Саме через паралельні порти мікроконтролер зв'язується з зовнішнім світом.

Вхідний порт (порт вводу) у найпростішому випадку - це паралельний регістр, у який процесор може записувати інформацію. Вихідний порт (порт виводу) зазвичай просто однонапрямлений буфер, через який процесор може читати інформацію із зовнішнього пристрою. Саме такі порти показані для прикладу на Рис. 8.1. Порт може бути і двонапрямленим (вхідним/вихідним). У цьому випадку процесор пише інформацію в зовнішній пристрій і читає інформацію з зовнішнього пристрою за тою самою адресою в адресному просторі системи. Вхідні і вихідні лінії для зв'язку з зовнішнім пристроєм при цьому можуть бути об'єднані порозрядно, утворюючи двонапрямлені лінії.

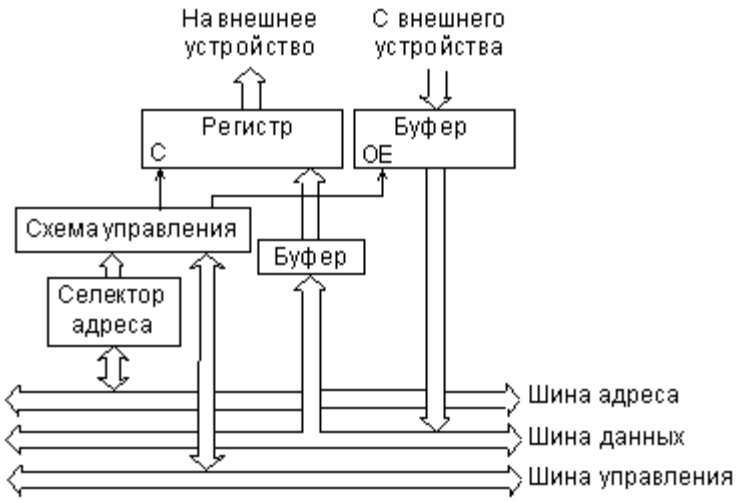


Рис. 8.1. Структура найпростішого пристрою вводу/виводу.

При звертанні з боку магістралі селектор адреси розпізнає адресу, яка приписана даному пристрою вводу/виводу. Схема управління видає внутрішні строби обміну у відповідь на магістральні строби обміну. Вхідний буфер даних забезпечує електричне узгодження шини даних з цим пристроєм (буфер може й бути відсутнім). Дані із шини даних записуються в регістр за сигналом С і видаються на зовнішній пристрій. Вихідний буфер даних передає вхідні дані з зовнішнього пристрою на шину даних магістралі в циклі читання з порту.

Складніші пристрої вводу/виводу (пристрої узгодження) мають у своєму складі внутрішню буферну оперативну пам'ять і навіть можуть мати мікроконтролер, на який покладене виконання функцій обміну з зовнішнім пристроєм.

Кожному пристрою вводу/виводу призначається своя адреса в адресному просторі мікропроцесорної системи. Дублювання адрес повинно бути виключене, за цим повинні стежити виробник і користувач мікропроцесорної системи.

Пристрої вводу/виводу крім програмного обміну можуть також підтримувати режим обміну за перериваннями. У цьому випадку вони перетворять сигнал запиту на переривання, що надходить від зовнішнього пристрою, в сигнал запиту переривання, необхідний для даної магістралі (чи в послідовність сигналів при векторному перериванні). Якщо потрібно використовувати режим ПДП, пристрій вводу/виводу повинен видати сигнал запиту ПДП на магістраль і забезпечити роботу в циклах ПДП, прийнятих для даної магістралі.

У складі мікропроцесорних систем, як правило, виділяються три спеціальні групи пристроїв вводу/виводу:

- пристрої **інтерфейсу** користувача (вводу інформації користувачем і виводу інформації для користувача);
- пристрої вводу/виводу для тривалого збереження інформації;
- таймерні пристрої.

До пристроїв вводу для інтерфейсу користувача відносяться контролери клавіатури, тумблерів, окремих кнопок, миші, трекболу, джойстика та ін. До пристроїв виводу для інтерфейсу користувача відносяться контролери світлодіодних індикаторів, табло, рідкокристалічних, плазмових і електронно-променевих екранів та ін. У найпростіших випадках управляючих контролерів чи мікроконтролерів ці засоби можуть бути відсутніми. У складних мікропроцесорних системах вони є обов'язково. Роль зовнішнього пристрою в даному випадку грає людина.

Пристрої вводу/виводу для тривалого збереження інформації забезпечують узгодження мікропроцесорної системи з дисководами (компакт-дисків чи магнітних дисків), а також з накопичувачами на магнітній стрічці. Застосування таких пристроїв істотно збільшує можливості мікропроцесорної системи у відношенні збереження виконуваних програм і накопичення масивів даних. У найпростіших контролерах ці пристрої відсутні.

Таймерні пристрої відрізняються від інших пристроїв вводу/виводу тим, що вони можуть не мати зовнішніх виводів для підключення до зовнішніх пристроїв. Ці пристрої призначені для того, щоб мікропроцесорна система могла витримувати задані часові інтервали, стежити за реальним часом, рахувати імпульси і т.д. В основі будь-якого таймера лежить кварцовий тактовий генератор і багаторозрядні двійкові лічильники, які можуть перезапускати один одного. Процесор може записувати в таймер коефіцієнти ділення тактової частоти, кількість відлічуваних імпульсів, задавати режим роботи лічильників таймера, а читає процесор вихідні коди лічильників. У принципі виконати практично усі функції таймера можна і програмним шляхом, тому іноді таймери в системі відсутні. Але включення в систему таймера дозволяє вирішувати більш складні задачі і будувати більш ефективні алгоритми.

Ще один важливий клас пристроїв вводу/виводу - це пристрої для підключення до інформаційних мереж (локальних і глобальних). Ці пристрої поширені не так широко, як пристрої трьох перерахованих раніше груп, але їхнє значення з кожним роком стає усе більшим. Зараз засоби зв'язку з інформаційними мережами вводяться іноді навіть у прості контролери.

Іноді пристрої вводу/виводу забезпечують узгодження з зовнішніми пристроями за допомогою аналогових сигналів. Це буває дуже зручно, тому до складу деяких мікроконтролерів навіть уводять внутрішні ЦАП і АЦП.

8.1. Порти вводу/виводу

Кожен МК має деяку кількість ліній вводу/виводу, що об'єднані в багаторозрядні (частіше 8-розрядні) паралельні **порти** вводу/виводу. У пам'яті МК кожному порту вводу/виводу відповідає своя адреса регістра даних. Звертання до регістра даних порту вводу/виводу відбувається тими ж командами, що і звертання до пам'яті даних. Крім того, у багатьох МК окремі розряди портів можуть бути опитані або встановлені командами бітового процесора.

У залежності від реалізованих функцій розрізняють наступні типи паралельних портів:

- однонапрямлені порти, призначені тільки для вводу або тільки для виводу інформації;
- двонапрямлені порти, напрямок передачі яких (ввід або вивід) визначається в процесі ініціалізації МК;
- порти з альтернативною функцією (мультиплексовані порти). Окремі лінії цих портів використовуються спільно з вмонтованими периферійними пристроями МК, такими як **таймери**, АЦП, контролери послідовних інтерфейсів;
- порти з програмно керованою схемотехнікою вхідного/вихідного буфера.

Порти виконують роль пристроїв часового узгодження функціонування МК і об'єкта управління, які, в загальному випадку, працюють асинхронно. Розрізняють три типи алгоритмів обміну інформацією між МК і зовнішнім пристроєм через паралельні порти вводу/виводу:

- режим простого програмного вводу/виводу;
- режим уводу/виводу зі стробом;
- режим уводу/виводу з повним набором сигналів підтвердження обміну.

Типова схема двонапрямленого порту вводу/виводу МК наведена на Рис. 8.2.

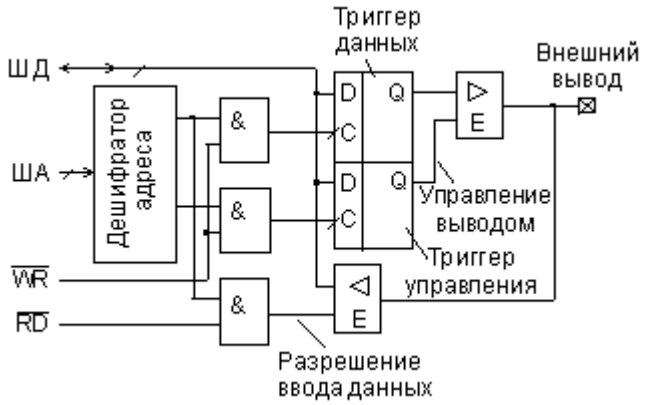


Рис. 8.2. Типова схема двонапрямленого порту вводу/виводу МК.

Тригер управління дозволяє вивід даних на зовнішній вивід. У сучасних МК, як правило, забезпечується індивідуальний доступ до тригерів даних і управління, що дозволяє використовувати кожну лінію незалежно в режимі **вводу** або **виводу**.

Необхідно звернути особливу увагу на те, що при введенні даних зчитується значення сигналу, що надходить на зовнішній вивід, а не вміст тригера даних. Якщо до зовнішнього виводу МК підключені виходи інших пристроїв, то вони можуть установити свій рівень вихідного сигналу, який і буде зчитаний замість очікуваного значення тригера даних.

Іншим розповсюдженим варіантом схемотехнічної організації порту вводу/виводу є вивід з "відкритим витоком", який ще називають

"квазідвонапрямленим". Така організація виводу дозволяє створювати шини з об'єднанням пристроїв за схемою "монтажне І".

8.2. Таймери і процесори подій

Більшість задач управління, які реалізуються за допомогою МК, вимагають виконання їх у реальному часі. Під цим розуміється здатність системи одержувати інформацію про стан керованого об'єкта, виконувати необхідні розрахункові процедури і видавати керуючі сигнали протягом інтервалу часу, достатнього для бажаної зміни стану об'єкта .

Покладати функції формування управління в реальному масштабі часу тільки на центральний процесор неефективно, тому що це займає ресурси, необхідні для розрахункових процедур. Тому в більшості сучасних МК використовується апаратна підтримка роботи в реальному часі з використанням таймера (таймерів).

Модулі таймерів служать для прийому інформації про час настання тих або інших подій від зовнішніх датчиків подій, а також для формування керуючих впливів у часі.

Модуль таймера 8-розрядного МК являє собою 8-ми або 16-розрядний **лічильник** зі схемою управління. Схемотехнікою МК звичайно передбачається можливість використання таймера в режимі лічильника зовнішніх подій, тому його часто називають таймером/лічильником. Структура типового 16-розрядного таймера/лічильника в складі МК наведена на Рис. 8.3.

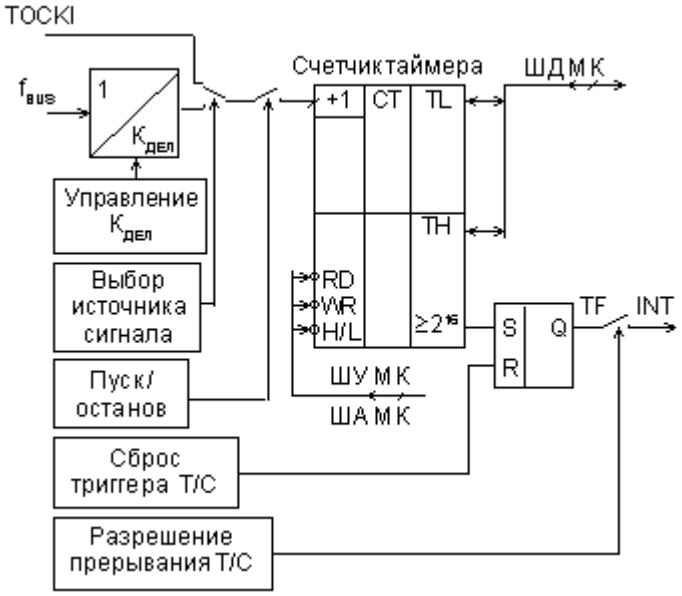


Рис. 8.3. Структура модуля таймера/лічильника.

У пам'яті МК 16-розрядний лічильник відображається двома регістрами: TH - старший байт лічильника, TL - молодший байт. Регістри доступні для читання і для запису. Напрямок відліку - тільки прямий, тобто при надходженні вхідних імпульсів уміст лічильника інкрементується. У залежності від настроювання лічильник може використовувати одне з джерел вхідних сигналів:

- імпульсну послідовність з виходу керованого дільника частоти f_{BUS};
- сигнали зовнішніх подій, що надходять на вхід TOCKI контролера.

У першому випадку говорять, що лічильник працює в режимі таймера, у другому - у режимі лічильника подій. При переповненні лічильника встановлюється в "одиницю" тригер переповнення TF, що генерує запит на **переривання**, якщо переривання від таймера дозволені. Пуск і зупинка таймера можуть здійснюватися тільки під управлінням програми. Програмним способом можна також установити старший і молодший біти лічильника в довільний стан або прочитати поточний код лічильника.

Розглянутий "класичний" модуль таймера/лічильника широко застосовується в різних моделях простих МК. Він може використовуватися для виміру часових інтервалів і формування послідовності імпульсів. Основними недоліками "класичного" таймера/лічильника є:

- втрати часу на виконання команд пуску і зупинки таймера, що створює помилки при вимірюванні часових інтервалів і обмежує мінімальну тривалість вимірюваних інтервалів часу одиницями мс.;
- складності при формуванні часових інтервалів (міток часу), відмінних від періоду повного коефіцієнта відліку, рівного $(K_{діл}/f_{BUS}) \cdot 2^{16}$;
- неможливість одночасного обслуговування (вимірювання або формування імпульсного сигналу) відразу декількох каналів.

Перші із двох перерахованих недоліків були усунуті в удосконаленому модулі таймера/лічильника, який застосовується в МК сімейства MCS-51 (Intel). Додаткова логіка рахункового входу дозволяє тактовим імпульсам надходити на вхід лічильника, якщо рівень сигналу на одній з ліній уводу дорівнює "1". Таке рішення підвищує точність виміру часових інтервалів, тому що запуск і зупинка таймера відбувається апаратно. Також в удосконаленому таймері реалізований режим перезавантаження лічильника довільним кодом у момент переповнення. Це дозволяє формувати часові послідовності з періодом, відмінним від періоду повного коефіцієнта відліку.

Однак ці удосконалення не усувають головного недоліку модуля "класичного" таймера - одноканального режиму роботи. Удосконалювання підсистеми реального часу МК ведеться за наступними напрямками:

- збільшення кількості модулів таймерів/лічильників. Цей шлях характерний для фірм, що випускають МК зі структурою MCS-51, а також для МК компаній Mitsubishi і Hitachi;
- модифікація структури модуля таймера/лічильника, при якій збільшення кількості каналів досягається не за рахунок збільшення кількості лічильників, а за рахунок уведення додаткових апаратних засобів вхідного захоплення (input capture - IC) і вихідного порівняння (output compare - OC). Такий підхід використовується, зокрема, у МК компанії Motorola.

Принцип дії каналу вхідного захоплення таймера/лічильника ілюструє Рис. 8.4.

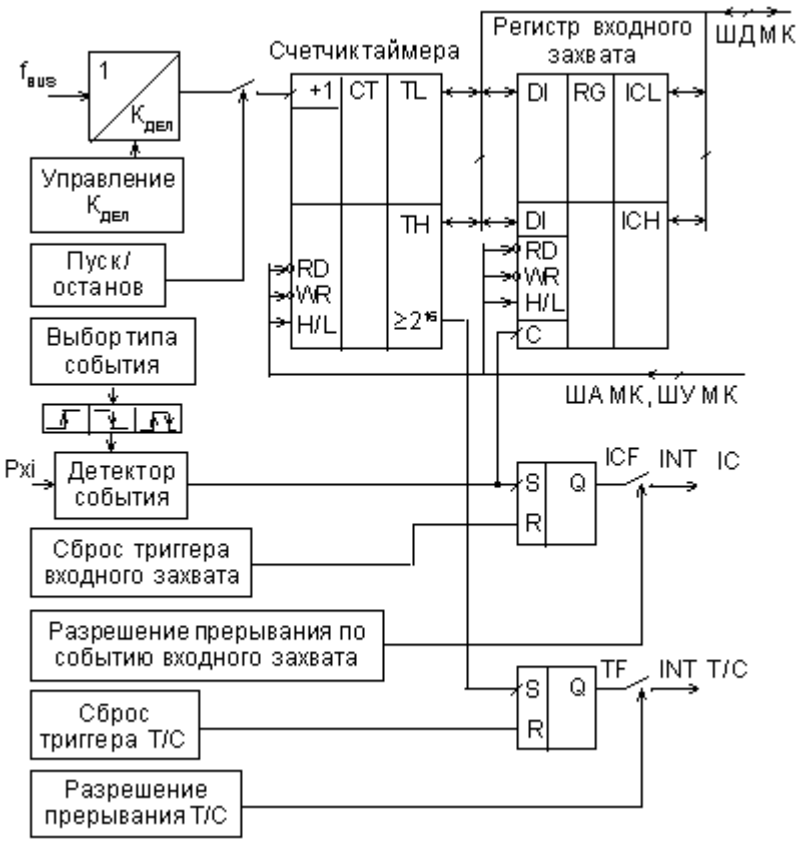


Рис. 8.4. Структурна схема каналу вхідного захоплення таймера.

Схема детектора події "спостерігає" за рівнем напруги на одному з входів МК. Найчастіше це одна з ліній порта вводу/виводу. При зміні рівня логічного сигналу з "0" на "1" і навпаки виробляється строб запису, і поточний стан лічильника таймера записується в 16-розрядний регістр вхідного захоплення. Описану дію в мікропроцесорній техніці називають подією захоплення. Передбачено можливість вибору типу сигналу на вході, і це сприймається як подія:

- позитивний (передній) фронт сигналу;
- негативний (задній) фронт сигналу;
- будь-яка зміна логічного рівня сигналу.

Вибір типу події захоплення встановлюється в процесі ініціалізації таймера і може неодноразово змінюватися в ході виконання програми. Кожна подія захоплення приводить до установки в "1" тригера вхідного захоплення і появі на його виході прапора (ознаки) вхідного захоплення ICF. Стан тригера вхідного захоплення може бути зчитаний програмно, а якщо переривання за подією захоплення дозволені - формується запит на переривання INT IC.

Використання режиму вхідного захоплення дозволяє виключити помилки вимірювання вхідного інтервалу часу, пов'язані з часом переходу до підпрограми обробки переривання, тому що копіювання поточного стану лічильника здійснюється апаратними, а не програмними засобами. Однак час переходу на підпрограму обробки переривання накладає обмеження на тривалість вимірюваного інтервалу часу, тому що передбачається, що друга

подія захоплення відбудеться пізніше, ніж код першої події буде зчитаний МК.

Структура апаратних засобів каналу вихідного порівняння наведена на Рис. 8.5.

Цифровий компаратор безупинно порівнює поточний код лічильника таймера з кодом, що записаний у 16-розрядному регістрі вихідного порівняння. У момент рівності кодів на одному з виходів МК (Pxj на Рис. 8.5) установлюється заданий рівень логічного сигналу. Зазвичай передбачено три типи зміни сигналу на виході Pxj у момент події вихідного порівняння:

- установка високого логічного рівня;
- установка низького логічного рівня;
- інвертування сигналу на виході.

При настанні події порівняння встановлюються в "1" тригер вихідного порівняння і відповідна йому ознака (прапорець) вихідного порівняння OCF. Аналогічно режиму вхідного захоплення стан тригера вихідного порівняння може бути зчитаний програмно, а якщо переривання за подією порівняння дозволені - формується запит на переривання INT OC.

Режим вихідного порівняння призначений, насамперед, для формування часових інтервалів заданої тривалості. Тривалість сформованого часового інтервалу визначається тільки різницею кодів, що завантажуються послідовно в регістр вихідного порівняння, і не залежить від програмного забезпечення МК. Час, необхідний для запису нового значення коду в регістр каналу порівняння, обмежує мінімальну тривалість формованого часового інтервалу.

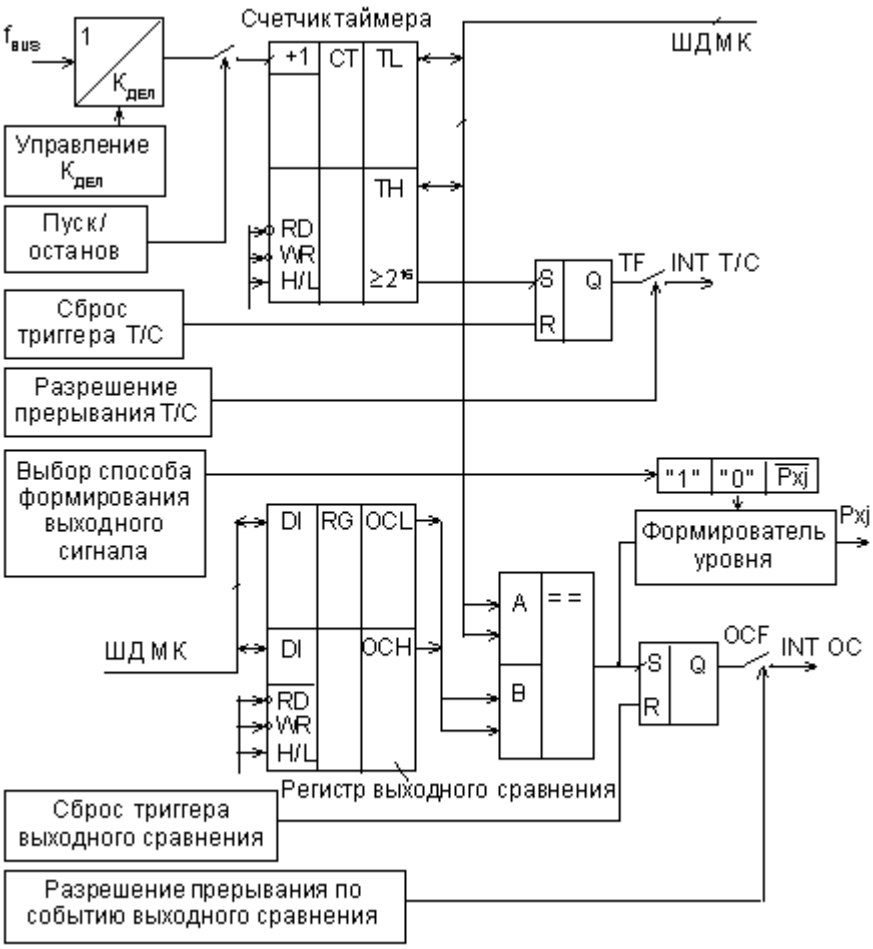


Рис. 8.5. Структурна схема каналу вихідного порівняння таймера.

Модулі удосконаленого таймера використовуються в складі МК у різних модифікаціях. При цьому кількість каналів вхідного захоплення і вихідного порівняння в модулі може бути різною. Так, у МК сімейства HC05 фірми Motorola типовими рішеннями є модулі 1IC+1OC або 2IC+2OC, а модуль таймера в складі МК тільки один. У ряді модулів канали можуть бути довільно набудовані на функцію вхідного захоплення або вихідного порівняння за допомогою ініціалізації. Лічильник модуля удосконаленого таймера може не мати функції програмної зупинки. У цьому випадку стан лічильника не можна синхронізувати з яким-небудь моментом роботи МК, і такий лічильник характеризується як такий, що вільно рахує (free counter).

Апаратні засоби удосконаленого таймера дозволяють вирішити багато задач управління в реальному часі. Однак у міру росту складності алгоритмів управління чітко виявляються обмеження модулів удосконаленого таймера, а саме:

- недостатня кількість каналів захоплення і порівняння, що належать одному лічильнику часової бази. Це не дозволяє сформувати синхронізовані між собою багатоканальні імпульсні послідовності;
- однозначно визначена конфігурація каналу (або захоплення або порівняння) часто не задовольняє потреби розв'язуваної задачі;

- формування сигналів за методом широтно-імпульсної модуляції (ШІМ) вимагає програмної підтримки, що знижує максимально досяжну частоту вихідного сигналу.

Тому наступним етапом розвитку модулів підсистеми реального часу МК стали модулі **процесорів подій**. Уперше модулі процесорів подій були використані компанією Intel у МК сімейства 8x51Fx. Цей модуль одержав назву програмованого лічильного масиву (Programmable Counter Array - PCA).

PCA забезпечує більш широкі можливості роботи в реальному масштабі часу й у меншій мірі витрачає ресурси центрального процесора, ніж стандартний і удосконалений таймер/лічильник. До переваг PCA також можна віднести більш просте програмування і більш високу точність. Приміром, PCA може забезпечити краще часову роздільну здатність, ніж таймери 0, 1 і 2 МК сімейства MCS-51, тому що лічильник PCA здатний працювати з тактовою частотою, утроє більшою, ніж у цих таймерів. PCA також може рішати багато задач, виконання яких з використанням таймерів вимагає додаткових апаратних витрат (наприклад, визначення фазового зсуву між імпульсами чи генерація ШІМ-сигналу). PCA складається з 16-бітного таймера-лічильника і п'яти 16-бітних модулів порівняння-замикання, як показано на Рис. 8.6.

Таймер-лічильник PCA використовується як базовий таймер для функціонування всіх п'яти модулів порівняння-замикання. Вхід таймера-лічильника PCA може бути запрограмований на рахунок сигналів від наступних джерел:

- вихід дільника на 12 тактового генератора МК;
- вихід дільника на 4 тактові генератори МК;
- сигнал переповнення таймера 0;
- зовнішній вхідний сигнал на виводі ECI (P1.2).

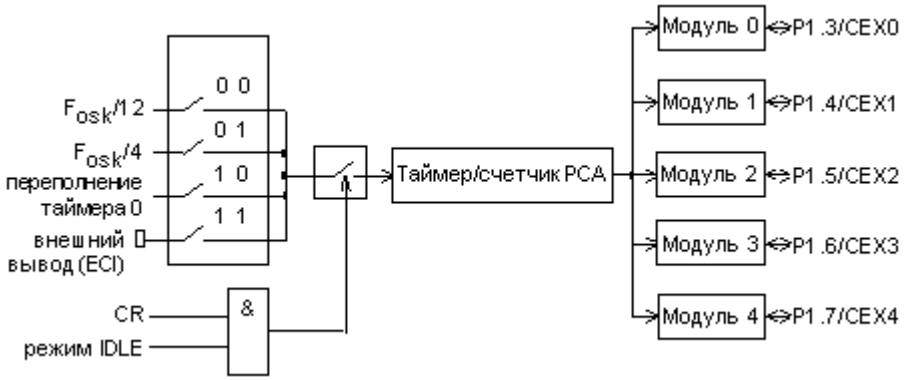


Рис. 8.6. Структура процесора подій МК сімейства Intel 8x51Fx.

Будь-який з модулів порівняння-замикання може бути запрограмований для роботи в наступних режимах:

- замикання за фронтом і/або спадом імпульсу на вході CEXi;
- програмованого таймера;
- високошвидкісного виходу;
- широтно-імпульсного модулятора.

Модуль 4 може бути також запрограмований як сторожовий таймер (Watchdog Timer - WDT).

Режим замикання за імпульсом на вході МК еквівалентний режимові вхідного захоплення (IC) удосконаленого таймера. Режими програмованого таймера і високошвидкісного виходу близькі за своїми функціональними можливостями до режиму вихідного порівняння (OC).

У режимі ШІМ на відповідному виводі МК формується послідовність імпульсів з періодом, рівним періодові базового таймера/лічильника PCA. Значення 8-розрядного коду, записане в молодший байт регістра-замка відповідного модуля задає скважність сформованого сигналу. При зміні коду від 0 до 255 скважність змінюється від 100% до 0,4%.

Режим ШІМ дуже простий з погляду програмного обслуговування. Якщо зміни скважності не передбачається, то досить один раз занести відповідний код у регістр даних модуля, проініціалізувати режим ШІМ, і імпульсна послідовність буде відтворюватися з заданими параметрами без втручання програми.

Призначення й особливості роботи сторожового таймера будуть розглянуті далі окремо.

При роботі модуля порівняння-замикання в режимі замикання, програмованого таймера або високошвидкісного виходу модуль може сформувати сигнал переривання. Сигнали від усіх п'яти модулів порівняння-замикання і сигнал переповнення таймера PCA поділяють один вектор переривання. Іншими словами, якщо переривання дозволені, то і сигнал переповнення таймера PCA і сигнал від кожного з модулів викликають ту саму підпрограму переривань. Вона повинна сама ідентифікувати джерело, яке її викликало.

Для роботи з зовнішніми пристроями таймер-лічильник PCA і модулі порівняння-замикання використовують виводи P1 порта МК. Якщо який-небудь вивід порта не використовується при роботі PCA, або PCA не задіяний, порт може застосовуватися стандартним способом.

Реалізований у 8x51FX PCA виявився настільки вдалим, що архітектура даних МК стала промисловим стандартом де-факто, а сам PCA багаторазово відтворювався в різних модифікаціях мікроконтролерів різних фірм.

Тенденція розвитку підсистеми реального часу сучасних МК знаходить своє відображення в збільшенні числа каналів процесорів подій і розширенні їхніх функціональних можливостей.

Описані вище модулі складають так званий базовий комплект МК і входять до складу будь-якого сучасного контролера. Очевидна необхідність включення до складу МК додаткових модулів, склад і можливості яких визначаються конкретною розв'язуваною задачею. Серед таких додаткових модулів необхідно, насамперед, відзначити:

- модулі послідовного вводу/виводу даних;
- модулі аналогового вводу/виводу.

8.4. Модулі послідовного вводу/виводу

Наявність у складі 8-розрядного МК модуля контролера послідовного вводу/виводу стало останнім часом звичайним явищем. Задачі, які розв'язуються засобами модуля контролера послідовного вводу/виводу, можна розділити на три основні групи:

- зв'язок вбудованої мікроконтролерної системи із системою управління верхнього рівня, наприклад, з персональним комп'ютером. Найчастіше для цієї мети використовуються інтерфейси RS-232C і RS-485;
- зв'язок із зовнішніми стосовно МК периферійними ІС, а також з датчиками фізичних величин з послідовним виходом. Для цієї мети використовуються інтерфейси I^2C, SPI, а також нестандартні протоколи обміну;
- інтерфейс зв'язку з локальною мережею в мультимікроконтролерних системах. У системах з кількістю МК до п'яти зазвичай використовуються мережі на основі інтерфейсів I^2C, RS-232C і RS-485 із власними мережевими протоколами високого рівня. У складніших системах усе популярнішим стає протокол CAN.

З погляду організації обміну інформацією згадані типи інтерфейсів послідовного зв'язку відрізняються режимом передачі даних (синхронний або асинхронний), форматом кадру (кількість біт у посилці при передачі байта корисної інформації) і часовими діаграмами сигналів на лініях (рівні сигналів і положення фронтів при перемиканнях).

Кількість ліній, якими відбувається передача в послідовному коді, зазвичай дорівнює двом (I^2C, RS-232C, RS-485) або трьом (SPI, деякі нестандартні протоколи). Дана обставина дозволяє спроектувати модулі контролерів послідовного обміну таким чином, щоб з їх допомогою на апаратному рівні можна було реалізувати кілька типів послідовних інтерфейсів. При цьому режим передачі (синхронний або асинхронний) і формат кадру підтримуються на рівні логічних сигналів, а реальні фізичні рівні сигналів для кожного інтерфейсу одержують за допомогою спеціальних ІС, які називають прийомопередавачами, конверторами, трансиверами.

Серед різних типів вмонтованих контролерів послідовного обміну, які входять до складу тих чи інших 8-розрядних МК, склався стандарт "де-факто" - модуль UART (Universal Asynchronous Receiver and Transmitter). UART - це універсальний асинхронний прийомопередавач. Однак більшість модулів UART, крім асинхронного режиму обміну, здатні також реалізувати режим синхронної передачі даних.

Не усі виробники МК використовують термін UART для позначення типу модуля контролера послідовного обміну. Так, у МК фірми Motorola модуль асинхронної прийомопередачі, що підтримує ті ж режими асинхронного обміну, що і UART, прийнято називати SCI (Serial Communication Interface). Слід зазначити, що модуль типу SCI переважно реалізує тільки режим асинхронного обміну, тобто його функціональні можливості вужчі у порівнянні з модулями типу UART. Однак бувають і винятки: під тим же

ім'ям SCI у МК MC68HC705B16 ховається модуль синхронно-асинхронної передачі даних.

Модулі типу UART в асинхронному режимі роботи дозволяють реалізувати протокол обміну для інтерфейсів RS-232C, RS-422A, RS-485, у синхронному режимі - нестандартні синхронні протоколи обміну, і в деяких моделях - SPI. У МК фірми Motorola традиційно передбачені два модулі послідовного обміну: модуль SCI з можливістю реалізації тільки протоколів асинхронної прийомопередачі для інтерфейсів RS-232C, RS-422A, RS-485 і модуль контролера синхронного інтерфейсу в стандарті SPI.

Протоколи інтерфейсів локальних мереж на основі МК (I^2C і CAN) відрізняє більш складна логіка роботи. Тому контролери CAN інтерфейсу завжди виконуються у виді самостійного модуля. Інтерфейс I^2C з можливістю роботи як у ведучому, так і веденому режимі, також підтримується спеціальним модулем (модуль **послідовного порту** в МК 89C52 фірми Philips). Але якщо реалізується тільки ведений режим I^2C, то в МК PIC16 фірми Microchip він успішно поєднується з SPI: настроювання того самого модуля на один із протоколів здійснюється шляхом ініціалізації.

Останнім часом з'явилася велика кількість МК з убудованими модулями контролерів CAN і модулями універсального послідовного інтерфейсу периферійних пристроїв USB (Universal Serial Bus). Кожен з цих інтерфейсів має досить складні протоколи обміну, для ознайомлення з якими варто звернутися до спеціальної літератури.

8.5. Модулі аналогового вводу/виводу

Необхідність прийому і формування аналогових сигналів вимагає наявності в МК модулів аналогового вводу/виводу.

Найпростішим пристроєм аналогового вводу в МК є вмонтований компаратор напруги. Компаратор порівнює вхідну аналогову напругу з опорним потенціалом V_{REF} і встановлює на виході логічну "1", якщо вхідна напруга більша за опорну. Компаратори зручніше за усе використовувати для контролю визначеного значення вхідної напруги, наприклад, у термостатах. У комбінації з зовнішнім генератором лінійно-змінної напруги, убудований компаратор дозволяє реалізувати на МК інтегруючий аналого-цифровий перетворювач (**АЦП**).

Однак ширші можливості для роботи з аналоговими сигналами дає АЦП, вмонтований у МК. Найчастіше він реалізується у виді модуля багатоканального АЦП, призначеного для вводу в МК аналогових сигналів з датчиків фізичних величин і перетворення цих сигналів у двійковий код. Структурна схема типового модуля АЦП наведена на Рис. 8.7.

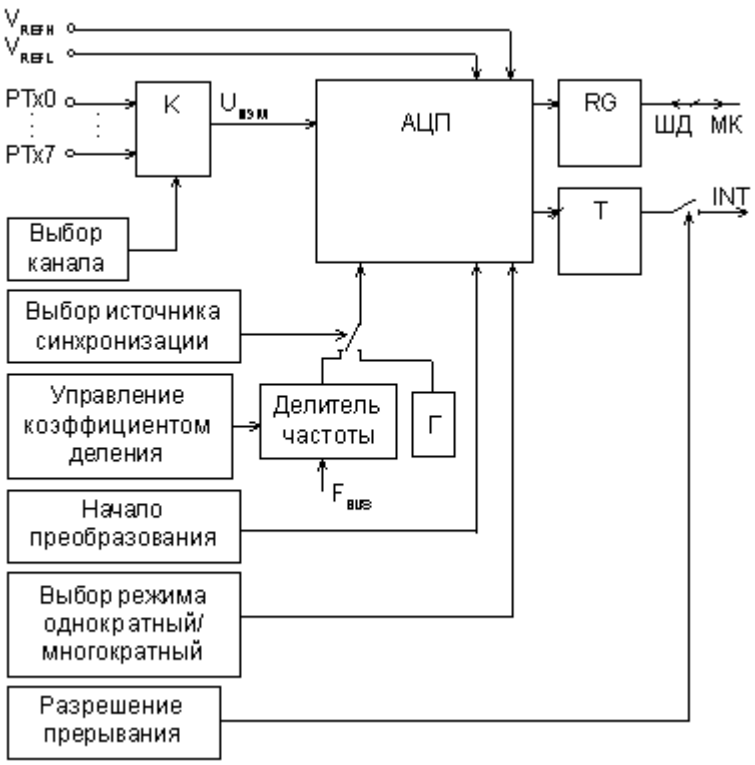


Рис. 8.7. Структура модуля АЦП.

Багатоканальний аналоговий комутатор К служить для підключення одного з джерел аналогових сигналів (PTx0...PTx7) до входу АЦП. Вибір джерела сигналу для перетворення здійснюється за допомогою запису номера каналу комутатора у відповідні розряди регістра управління АЦП.

Два виводи модуля АЦП використовуються для задавання опорної напруги $U_{оп}$: V_{REFH} - верхня межа $U_{оп}$, V_{REFL} - нижня межа. Різниця потенціалів на входах V_{REFH} і V_{REFL} і складає $U_{оп}$. Роздільна здатність АЦП складає $U_{оп}/2^n$, де n - кількість розрядів у слові результату. Максимальне значення опорної напруги, як правило, дорівнює напрузі живлення МК. Якщо вимірювана напруга $U_{вим} > V_{REFH}$, то результат перетворення буде дорівнює FF, код 00 відповідає напругам $U_{вим}$ J V_{REFL}. Для досягнення максимальної точності виміру варто вибрати максимально допустиме значення $U_{оп}$. У цьому випадку напруга зсуву нуля вхідного буфера і нелінійність передаточної характеристики АЦП будуть вносити відносно малі похибки.

Власне аналого-цифровий перетворювач виконаний за методом послідовного наближення. Практично у всіх моделях 8-розрядних МК розрядність АЦП також складає 8 розрядів. Відповідно, формат подання результатів виміру АЦП - однобайтовий. Виняток складають тільки модулі АЦП мікроконтролерів для управління перетворювачами частоти для електроприводів, роздільна здатність яких дорівнює 10 розрядам. Два молодших розряди результату одержують за допомогою додаткового ємнісного дільника, не зв'язаного з регістром послідовного наближення.

Тривалість такту перетворення задає генератор синхронізації: один цикл дорівнює двом періодам частоти генератора t_{ADC}. Час перетворення для

типових модулів АЦП мікроконтролерів складає від одиниць до десятків мікросекунд.

Джерелом синхронізації модуля АЦП може служити вмонтований RC-генератор (Г) або імпульсна послідовність тактування міжмодульних магістралей МК. У першому випадку частота синхронізації АЦП обов'язково виявиться оптимальною, тобто такою, яка рекомендується в технічному описі. В другому випадку обрана з інших причин f_{BUS} може виявитися невідповідною для модуля АЦП. На цей випадок у складі деяких модулів передбачений програмований дільник частоти f_{BUS}.

Момент завершення кожного циклу перетворення відзначається встановленням тригера готовності даних. Якщо переривання від модуля АЦП дозволені, то генерується запит на переривання. Як правило, читання регістра результату скидає тригер готовності.

Більшість модулів АЦП мають тільки режим програмного запуску: встановлення одного з бітів регістра режиму запускає чергове вимірювання. Найуніверсальніші модулі АЦП мають також режим автоматичного запуску, при якому після завершення одного циклу перетворення негайно починається наступний. Однак дані вимірювання кожного циклу повинні бути зчитані програмним способом.

Цифро-аналогові перетворювачі в складі МК є великою рідкістю. Функція цифро-аналогового перетворювача реалізується засобами модуля програмованого таймера в режимі ШІМ. На одному з виводів МК формується високочастотна імпульсна послідовність з регульованою тривалістю імпульсу. Отриманий сигнал згладжується фільтром нижніх частот на операційному підсилювачі. Роздільна здатність такого **ЦАП** визначається дискретністю регулювання коефіцієнта заповнення в режимі ШІМ.

ТЕМА 9. ОДНОКРИСТАЛЬНІ МІКРО-ЕОМ.

Мікроконтролери сімейств **PIC** (Peripheral Interface Controller) компанії Microchip поєднують усі передові технології мікроконтролерів: електрично програмовані користувачем ППЗП, мінімальне енергоспоживання, високу продуктивність, добре розвинуту RISC-архітектуру, функціональну закінченість і мінімальні розміри. Широка номенклатура виробів забезпечує використання мікроконтролерів у пристроях, призначених для різноманітних сфер застосування.

Перші мікроконтролери компанії Microchip PIC16C5x з'явилися наприкінці 1980-х років і завдяки своїй високій продуктивності і низької вартості склали серйозну конкуренцію 8-розрядним МК із CISC-архітектурою, які виготовлялися на той час.

Висока швидкість виконання команд у PIC-контролерах досягається за рахунок використання двохшинної гарвардської архітектури замість традиційної одношинної фон-нейманівської. Гарвардська архітектура базується на наборі **регістрів** з розділеними шинами й адресними просторами для команд і даних. Усі ресурси мікроконтролера, такі як**порти** вводу/виводу, комірки пам'яті і **таймер**, являють собою фізично реалізовані апаратні регістри.

Мікроконтролери PIC містять RISC-процесор із симетричною системою команд, що дозволяє виконувати операції з будь-яким регістром, використовуючи довільний **метод адресації**. Користувач може зберігати результат операції в самому регістрі-акумуляторі або в другому регістрі, який використовується для операції.

В даний час компанія Microchip випускає п'ять основних сімейств 8-розрядних RISC-мікроконтролерів, сумісних знизу нагору за програмним кодом:

- **PIC12CXXX** - сімейство мікроконтролерів, що випускаються в мініатюрному 8-вивідному виконанні. Ці мікроконтролери випускаються як з 12-розрядною (33 команди), так і з 14-розрядною (35 команд) системою команд. Містять вмонтований тактовий генератор, таймер/лічильник, сторожовий таймер, схему управління перериваннями. У складі сімейства є мікроконтролери з убудованим 8-розрядним чотирьохканальним АЦП. Здатні працювати при напрузі живлення до 2,5 В;
- **PIC16C5X** - базове сімейство мікроконтролерів з 12-розрядними командами (33 команди), що випускається в 18-, 20- і 28-вивідних корпусах. Являють собою прості недорогі мікроконтролери з мінімальною периферією. Здатність працювати при малій напрузі живлення (до 2 В) робить їх зручними для застосування в переносних конструкціях. До складу сімейства входять мікроконтролери підгрупи PIC16HV5XX, здатні працювати безпосередньо від батареї в діапазоні напруг живлення до 15 В;
- **PIC16CXXX** - сімейство мікроконтролерів середнього рівня з 14-розрядними командами (35 команд). Найбільш численне сімейство, що поєднує мікроконтролери з різноманітними периферійними пристроями, до

складу яких входять аналогові компаратори, аналогово-цифрові перетворювачі, контролери послідовних інтерфейсів SPI, USART і I^2C, таймери-лічильники, модулі захоплення/порівняння, широтно-імпульсні модулятори, сторожові таймери, супервізорні схеми і так далі;

- **PIC17CXXX** - сімейство високопродуктивних мікроконтролерів з розширеною системою команд 16-розрядного формату (58 команд), що працюють на частоті до 33 Мгц, з об'ємом пам'яті програм до 16 Кслів. Крім великої периферії, 16-рівневого апаратного стека і векторної система переривань, майже всі мікроконтролери цього сімейства мають убудований апаратний множник 8x8, що виконує операцію множення за один машинний цикл. Є одними із самих швидкодіючих у класі 8-розрядних мікроконтролерів;
- **PIC18CXXX** - сімейство високопродуктивних мікроконтролерів з розширеною системою команд 16-розрядного формату (75 команд) і убудованим 10-розрядним АЦП, які працюють на частоті до 40 Мгц. Містять 31-рівневий апаратний стек, вмонтовану пам'ять команд до 32 Кслів і здатні адресувати до 4 Кбайт пам'яті даних і до 2 Мбайт зовнішньої пам'яті програм. Розширене RISC-ядро мікроконтролерів даного сімейства оптимізоване під використання нового Сі-компілятора.

Більшість PIC-контролерів випускаються з однократно програмованою пам'яттю програм (OTP), з можливістю внутрисхемного програмування або масочним ПЗП. Для мети налагодження пропонуються більш дорогі версії з ультрафіолетовим стиранням і Flash-пам'яттю. Повний список модифікацій PIC-контролерів, що випускаються, включає порядку п'ятисот найменувань. Тому продукція компанії перекриває майже весь діапазон застосувань 8-розрядних мікроконтролерів.

З програмних засобів налагодження найбільш відомі і доступні різні версії асемблерів, а також інтегроване програмне середовище MPLAB. Російські виробники програматорів і апаратних відлагоджувальнх засобів також приділяють увагу PIC-контролерам. Випускаються як спеціалізовані программатори, такі як PICPROG, що програмують майже весь спектр PIC-мікроконтролерів, так і універсальні: UNIPRO і CTEPX, які підтримують найбільш відомі версії PIC-контролерів.

Найбільш розповсюдженими сімействами PIC-контролерів є PIC16CXXX і PIC17CXXX.

5.1.2. Мікроконтролери сімейств PIC16CXXX і PIC17CXXX

Основним призначенням мікроконтролерів сімейств PIC16 і PIC17, як випливає з абревіатури PIC (Peripheral Interface Controller), є виконання інтерфейсних функцій. Цим підрозуміваєтся особливості їхньої архітектури:

- RISC-система команд, що характеризується малим набором одноадресних інструкцій (33, 35 або 58), кожна з яких має довжину в одне слово (12, 14 або 16 біт) і більшість виконується за один машинний цикл. У системі команд відсутні складні арифметичні команди (множення, ділення), гранично скорочений набір умовних переходів;

- висока швидкість виконання команд: при тактовій частоті 20 Мгц час машинного циклу складає 200 нс (швидкодія складає 5 млн. операцій/сек);
- наявність потужних драйверів (до 25 ма) на лініях портів вводу/виводу, що дозволяє підключати безпосередньо до них досить потужне навантаження, наприклад, світлодіоди.
- низька потужність споживання;
- орієнтація на цінову нішу гранично низької вартості, що визначає використання дешевих корпусів з малою кількістю виводів (8, 14, 18, 28), відмова від зовнішніх шин адреси і даних (крім PIC17C4X), використання спрощеного механізму **переривань** та апаратного (програмно недоступного) **стека**.

5.1.3. Особливості архітектури мікроконтролерів сімейства PIC16CXXX

Мікроконтролери сімейства PIC16CXXX, виконані за технологією HCMOS являють собою 8-розрядні мікроконтролери на основі RISC-процесора, виконані за гарвардською архітектурою. Мають вмонтований ПЗП команд об'ємом від 0,5 до 4 Кслів (розрядність слова команд дорівнює 12 - 14 біт). Пам'ять даних PIC-контролерів організована у виді реєстрового файлу об'ємом 32 - 128 байт, у якому від 7 до 16 регістрів відведено для управління системою та обміну даними з зовнішніми пристроями.

Одним з основних переваг цих пристроїв є дуже широкий діапазон напруг живлення (2 - 6 В). Струм споживання на частоті 32768 Гц складає менше 15 мкА, на частоті 4 МГц - 1 - 2 мА, на частоті 20 МГц 5 - 7 мА і у режимі мікроспоживання (режим SLEEP) - 1 - 2 мкА. Випускаються модифікації для роботи в трьох температурних діапазонах: від 0 до +70°С, від -40 до +85°С та від -40 до +125°С.

Кожен з контролерів містить універсальні (від 1 до 3) сторожові таймери, а також надійно збудовану систему ініціалізації при увімкненні живлення. Частота внутрішнього тактового генератора задається або кварцовим резонатором, або RC-ланкою у діапазоні 0 - 25 Мгц. PIC-контролери мають від 12 до 33 ліній цифрового вводу-виводу, причому кожна з них може бути незалежно настроєна на ввід або вивід.

У пристрій PIC16C64 входить широтно-імпульсний модулятор, за допомогою якого можна реалізувати ЦАП з роздільною здатністю до 16 розрядів. Тут є і послідовний двонапрямлений синхронно-асинхронний порт, що забезпечує можливість організації шини I^2C. Прилади PIC16C71 і PIC16C74 містять вмонтований багатоканальний 8-розрядний АЦП із пристроєм вибірки-збереження.

Крім пам'яті програм у PIC передбачено декілька перемичок, що індивідуально пропалюються, за допомогою яких можна на етапі програмування кристалу вибрати тип тактового генератора, відключити сторожовий таймер або систему ініціалізації, увімкнути захист пам'яті програм від копіювання, а також записати серійний номер кристала (16 біт).

З програмної точки зору PIC-контролер це 8-розрядний RISC-процесор з гарвардською архітектурою. Кількість команд невелика - від 33 до 35. Усі команди мають однакову довжину і, крім команд розгалуження, виконуються

за чотири періоди тактової частоти (на відмінність, наприклад, від 12 періодів для I87C51). Підтримуються безпосередній, непрямий і відносний методи адресації, можна ефективно керувати окремими бітами в межах усього реєстрового файлу. Стек реалізований апаратно. Його максимальна глибина складає два або вісім рівнів у залежності від типу контролера. Майже у всіх мікросхемах PIC є система переривань, джерелами яких можуть бути таймер і зовнішні сигнали. Система команд практично симетрична і, як наслідок, легка в освоєнні.

Застосування PIC-контролерів доцільне в нескладних приладах з обмеженим струмом споживання (автономні пристрої, прилади з живленням від телефонної лінії і т.п.). Завдяки малій кількості компонентів, використовуваних при побудові таких приладів, їхні розміри зменшуються, а надійність збільшується.

Типовим представником мікроконтролерів сімейства PIC16CXXX є мікроконтролери підгрупи PIC16F8X.

ТЕМА 10. ОРГАНІЗАЦІЯ АПАРАТНОГО І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МІКРОПРОЦЕСОРНИХ СИСТЕМ ДЛЯ ЕЛЕКТРОПОБУТОВОЇ ТЕХНІКИ.

Розробка програмного забезпечення є центральним моментом загального процесу проектування. Центр ваги функціональних властивостей сучасних цифрових систем знаходиться саме в програмних засобах.

Основним інструментом для професійної розробки програм є **асемблер**, що припускає деталізацію на рівні команд МК. Тільки асемблер дозволяє максимально використовувати ресурси кристала.

Для мікроконтролерів PIC випущена велика кількість різних засобів розробки. У даному розділі мова йтиме про засоби, наданих фірмою Microchip, які дуже ефективні і широко використовуються на практиці.

10.1. Асемблер MPASM

Асемблер MPASM являє собою інтегроване програмне середовище для розробки програмних кодів PIC мікроконтролерів усіх сімейств. Випускається фірмою Microchip у двох варіантах: для роботи під DOS і для роботи під Windows 95/98/NT. Асемблер MPASM може використовуватися як самостійно, так і в складі інтегрованого середовища розробки MPLAB. Він включає кілька програм: власне MPASM, MPLINK і MPLIB, причому кожна з них має власний інтерфейс.

Програма MPASM може використовуватися для двох цілей:

- генерації виконуваного (абсолютного) коду, призначеного для запису в МК за допомогою програматора;
- генерації переміщуваного об'єктного коду, що потім буде зв'язаний з іншими асембльованими чи компільованими модулями.

Код, який виконується, є для MPASM вихідним кодом за замовчуванням. При цьому всі змінні джерела повинні бути явно описані в тексті програми чи у файлі, що підключається за допомогою директиви INCLUDE <filename>. Якщо при асемблюванні не виявляється помилок, то генерується вихідний .hex-файл, що може бути завантажений у МК за допомогою програматора.

При використанні асемблера MPASM у режимі генерації переміщуваного об'єктного коду формуються об'єктні модулі, які можуть бути згодом об'єднані з іншими модулями за допомогою компоновника MPLINK. Програма-компоновник MPLINK перетворить переміщувані об'єктні коди в бінарний виконуваний код, прив'язаний до абсолютних адрес МК. Бібліотечна утиліта MPLIB дозволяє для зручності роботи згрупувати переміщувані об'єкти в один файл чи бібліотеку. Ці бібліотеки можуть бути зв'язані компоновником MPLINK у файл вихідного об'єктного коду асемблера MPASM.

Програми MPASM і MPLINK доступні через оболонку MPASM, тоді як MPLIB доступна тільки з свого командного рядка.

Вихідним файлом для асемблера MPASM за замовчуванням є файл із розширенням .ASM. Текст вихідного файлу повинен відповідати вимогам синтаксису, який наведено далі.

Асемблер MPASM може бути викликаний командним рядком:

MPASM [/<Option>[/<Option>...]] <file_name>

де /<Option> означає вибір режиму роботи асемблера в командному рядку; <file_name> - ім'я файлу для асемблювання.

Режими роботи асемблера, які обрані за замовчуванням, приведені в табл. 10.1.

Табл. 10.1. Режими роботи асемблера за замовчуванням.

Вибір	**Значення за замовчуванням**	**Опис**
?	N/A	Викликати допомогу
a	INHX8M	Генерувати абсолютний .COD і hex вихід безпосередньо з асемблера:
c	On	Вибрати/заборонити випадок чутливості
e	On	Вибрати/заборонити файл помилок
h	N/A	Відобразити панель допомоги MPASM
l	On	Вибрати/заборонити файл лістингу, який генерується з макроасемблера.
m	On	Викликати/заборонити макророзширення
o	N/A	Встановити шлях для об'єктних файлів /o<path>\object.file
p	None	Встановити тип процесора: /p<processor_type>
q	Off	Дозволити/Заборонити схований режим (заборонити вивід на екран)
r	Hex	Визначає тип числа за замовчуванням: /r<radix>
w	0	Визначає рівень діагностичних повідомлень у файлі лістингу /w<level>, де <level> може бути: 0 - повідомляти все, 1 - повідомляти про попередження і помилки, 2 - повідомляти тільки про помилки.
x	Off	Дозволити/заборонити перехресні посилання у файлі лістингу.

Тут і далі використовуються наступні угоди по використанню символів:

[ ] - для аргументів на вибір;

< > - для виділення спеціальних ключів <TAB>, <ESC> чи додаткового вибору;

| - для взаємовиключних аргументів (вибір АБО);

рядкові символи - для позначення типу даних.

Вибір за замовчуванням, наведений у табл. 10.1, може бути змінений командним рядком:

/<option> дозволяє вибір;

/<option>+ дозволяє вибір;

/<option> - забороняє вибір.

Вихідний асемблерний файл створюється з використанням будь-якого ASCII текстового редактора. Кожна лінія вихідного файлу може містити до чотирьох типів інформації:

- мітки (labels)
- мнемоніка (mnemonics)
- операнди (operands)
- коментар (comments)

Порядок і положення кожного типу має значення. Мітка повинна починатися в колонці номер один. Мнемоніка може починатися в колонку два чи далі. Операнди йдуть за мнемонікою. Коментар може слідувати за операндом, мнемонікою чи міткою чи може починатися в будь-якому стовпці, якщо в якості першого не порожнього символу використовується * чи ;.

Максимальна довжина рядка 255 символів.

Один чи кілька пробілів повинні відокремлювати мітку і мнемоніку чи мнемоніку й операнд(и). Операнди можуть відокремлюватися комами. Наприклад:

```
List    p=16C54, r=HEX
        ORG 0x1FF       ;Вектор скидання
        GOTO START      ;Повернення на початок
        ORG 0x000       ;Адреса початку виконання програми
START
        MOVLW 0x0A      ;Виконання програми PIC МК
        MOVLW 0x0B      ;Виконувати завжди
        GOTO START
        END
```

Мітки

У полі мітки розміщується символічне ім'я комірки пам'яті, у якій зберігається відзначений операнд. Усі мітки повинні починатися в колонці 1. За ними може слідувати двокрапка (:), пробіл, табуляція чи кінець рядка. Коментар може також починатися в колонці 1, якщо використовується одне з позначень коментарю.

Мітка може починатися з символу чи нижнього тире (_) і містити буквені символи, числа, нижні тире і знак питання. Довжина мітки може бути до 32 символів.

Мнемоніки

Мнемоніки це мнемонічні позначення команди, які безпосередньо транслюються в машинний код. Мнемоніки асемблерних інструкцій, директиви асемблера і макровиклики повинні починатися, принаймні, у колонці 2. Якщо є мітка на тій же лінії, вона повинна бути відділена від цієї мітки двокрапкою або одним чи більше пробілами або табуляцією.

Операнди

У цьому полі визначаються операнди (чи операнд), що беруть участь в операції. Операнди повинні бути відділені від мнемоніки одним або більше пробілами чи табуляцією. Операнди відокремлюються один від одного комами. Якщо операція вимагає фіксованого номера (числа) чи операндів, то все на лінії після операндів ігнорується. Коментарі дозволяються наприкінці лінії. Якщо мнемоніки дозволяють використовувати різну кількість операндів, кінець списку операндів визначається кінцем рядка чи коментарем.

Вирази використовуються в полі операнда і можуть містити константи, символи чи будь-які комбінації констант і символів, розділених арифметичними операторами. Перед кожною константою чи символом може стояти + чи - , що вказує на позитивний чи негативний вираз.

В асемблері MPASM використовуються наступні формати виразів:

- текстовий рядок;
- числові константи і Radix;
- арифметичні оператори і пріоритети;
- High / Low оператори.

Текстовий рядок - це послідовність будь-яких припустимих ASCII символів (у десятковому діапазоні від 0 до 127), поміщена в подвійні лапки. Рядок може мати будь-яку довжину в межах 132 стовпчиків. При відсутності обмеження рядка він вважається до кінця лінії. Якщо рядок використовується як буквений операнд, він повинен мати довжину в один символ, інакше буде помилка.

Числова константа це число, виражене в деякій системі числення. Перед константою може стояти + чи -. Проміжні значення в константах розглядаються як 32-розрядні цілі без знака.

MPASM підтримує наступні системи числення (представлення значень чи Radix): шіснадцядкову, десяткову, вісімкову, двійкову і символьну. За замовчуванням приймається шіснадяткова система. В Табл. 10.2 наведені різні системи числення.

Оператори - це арифметичні символи, подібні + і -, які використовуються при формуванні виразів. Кожен оператор має свій пріоритет. У загальному випадку пріоритет встановлюється зліва направо, а вирази в дужках оцінюються першими. У табл. 10.3 наведені позначення, описи і приклади застосування основних операторів MPASM.

Табл. 10.2. Системи числення (Radix).		
Тип	**Синтаксис**	**Приклад**
Десяткова	D'<цифри>' або .<цифри>	D'100' або .100
Шіснадцяткова	H'<цифри>' або 0x<цифри>	H'9f' або 0x9f
Вісімкова	O'<цифри>'	O'777'
Двійкова	B'<цифри>'	B'00111001'
Символьна	'<символ>' або A'<символ>'	"C" або A'C'

Табл. 10.3. Основні арифметичні оператори MPASM		
Оператор	**Опис**	**Приклад**
$	поточний лічильник команд	goto $ + 3
(	ліва дужка	1 + (d * 4)
)	права дужка	(lenght + 1) * 255
!	операція "НЕ" (логічна інверсія)	if ! (a - b)
~	доповнення	flags = ~ flags
-	інверсія (двійкове доповнення)	– 1 * lenght
High	виділити старший байт слова	movlw high llasid
Low	виділити молодший байт слова	movlw low (llasid + .251)
upper	виділити найбільший байт слова	movlw upper (llasid + .251)
*	множення	a = c * b
/	ділення	a = b / c
%	модуль	lenght = totall % 16
+	додавання	Tot_len = lenght * 8 + 1
–	віднімання	Entry_Son = (Tot – 1) / 8
<<	зсув вліво	Val = flags << 1
>>	зсув вправо	Val = flags >> 1
>=	більше або дорівнює	if ent >= num

>	більше	if ent > num
<	менше	if ent < num
<=	менше або дорівнює	if ent <= num
==	дорівнює	if ent == num
!=	не дорівнює	if ent != num
&	порозрядне "І"	flags = flags & err_bit
^	порозрядне "ВИКЛЮЧАЮЧЕ АБО"	flags = flags ^ err_bit
|	порозрядне "АБО"	flags = flags | err_bit
&&	логічне "І"	if (len == 512)&&(b == c)
||	логічне "АБО"	if (len == 512) || (b == c)
=	встановити рівним...	entry_index = 0
++	збільшити на 1 (інкремент)	i ++
—	зменшити на 1 (декремент)	i —

Оператори high, low і upper використовуються для одержання одного байта з багатобайтного значення, що відповідає мітці. Застосовуються для управління розрахунком крапок динамічного переходу при читанні таблиць і запису програм.

Оператори інкременту і декременту можуть застосовуватися до змінної тільки як єдиного оператора в рядку. Вони не можуть бути вбудованим фрагментом більш складного виразу.

Коментарі

Поле коментарю може використовуватися програмістом для текстового чи символьного пояснення логічної організації програми. Поле коментарю цілком ігнорується асемблером, тому в ньому можна застосовувати будь-які символи. Коментарі, які використовуються в рядку самі по собі, повинні починатися із символу коментарю (* чи ;). Коментарі наприкінці рядка повинні бути відділені від залишку рядка одним чи більше пробілами або табуляцією.

Розширення файлів, які використовуються MPASM і утилітами

Існує ряд розширень файлів, застосовуваних за замовчуванням MPASM і зв'язаними з ним утилітами. Призначення таких розширень наведені в табл. 10.4.

Табл. 10.4. Використовувані за замовчуванням призначення розширень файлів.	
Розширення	**Призначення**
.ASM	Вхідний файл асемблера для MPASM <source_name>.ASM
.OBJ	Вихідний файл переміщуваного об'єктного коду з <source_name>.OBJ
.LST	Вихідний файл лістингу, який генерируєтьcя асемблером MPASM чи MPLINK: <source_name>.LST
.ERR	Вихідний файл помилок з MPASM: <source_name>.ERR
.MAP	Вихідний файл розподілу пам'яті з MPASM: <source_name>.MAP
.HEX	Вихідний файл об'єктного коду в шіснадцятковому поданні з MPASM: <source_name).HEX
.HXL/.HXH	Вихідний файл об'єктного коду в шіснадцятковому поданні р роздільним поданням молодших і старших байтів: <source_name>.HXL, <source_name>.HXH
.LIB	Бібліотечний файл, створений MPLIB і прив'язаний компоновщиком MPLINK: <source_name>.LIB
.LNK	Вихідний файл компоновщика: <source_name>.LNK
.COD	Вихідний символьний файл чи файл відлагоджувальника. Формуються MPASM чи MPLINK: <source_name>.COD

Лістинг це текстовий файл у форматі ASCII, що містить машинні коди, згенеровані відповідно до кожної асемблерної команди, директивою асемблера чи макрокомандою вихідного файлу. Файл лістингу містить: ім'я продукту і версії, дату і час, номер сторінки вгорі кожної сторінки.

До складу лістингу входять також таблиця символів і карта використання пам'яті. У таблиці символів перераховуються всі символи, які є в програмі, і де вони визначені. Карта використання пам'яті дає графічну інформацію про витрату пам'яті МК.

Директиви мови

Директиви мови - це асемблерні команди, які зустрічаються у вихідному коді, але не транслюються прямо в коди, які виконуються. Вони використовуються асемблером при трактуванні мнемоніки вхідного файлу, розміщенні даних і формуванні файлу лістингу.

Існує чотири основних типи директив у MPASM:

- директиви даних;
- директиви лістингу;
- управляючі директиви;
- макро-директиви.

Директиви даних керують розподілом пам'яті і забезпечують доступ до символічних позначень даних.

Директиви лістингу керують лістингом файлу MPASM і форматом. Вони визначають специфікацію заголовків, генерацію сторінок і інші функції управління лістингом.

Директиви управління дозволяють зробити секціонування звичайного асемблерного коду.

Макро-директиви керують виконанням і розподілом даних у межах визначень макротіла.

Нижче наводиться опис деяких директив асемблера MPASM, використовуваних у даному навчальному посібнику.

CODE - початок секції об'єктного коду

Синтаксис:

[<label>] code [ROM address>]

Використовується при генерації об'єктних модулів. Повідомляє початок секції програмного коду. Якщо <label> не зазначена, секція буде названа .code Стартова адреса встановлюється рівною зазначеному значенню чи нулю, якщо адреса не була зазначена.

Приклад:

```
RESET code H'01FF'
    goto START
```

#DEFINE - визначити мітку заміни тексту

Синтаксис:

#define <name> [<string>]

Директива задає рядок <string>, що заміщає мітку <name> усякий раз, коли та буде зустрічатися у вихідному тексті. Символи, які визначені директивою #DEFINE, не можуть бути переглянуті **симулятором**. Використовуйте замість цієї директиви EQU.

Приклад:

```
#define length 20
#define control 0x19,7
#define position (X,Y,Z) (y-(2 * Z +X)).
test_label dw position(1, length, 512)
bsf control     ; установити в 1 біт 7 у f19
```

END - кінець програмного блоку

Синтаксис:

end

Визначає кінець програми. Після зупинки програми таблиця символів скидається у файл лістингу.

Приклад:

```
start
;код що виконується
;
end     ; кінець програми
```

EQU – визначити асемблерну константу

Синтаксис:

<label> equ <expr>

Тут <expr> – це правильний MPASM вираз. Значення виразу присвоюється мітці <label>.

Приклад:

four equ 4 ; присвоює чисельне значення мітці four

INCLUDE – увімкнути додатковий файл джерела

Синтаксис:

include <<include_file>>

include "<include_file>"

Обумовлений файл зчитується як джерело коду. По закінченні файлу, що включається, буде продовжуватися асемблювання джерела. Допускається до шести рівнів вкладеності. <include_file> може бути укладений у лапки чи кутові дужки. Якщо зазначено повний шлях до файлу, то пошук буде відбуватися тільки цим шляхом. У противному випадку порядок пошуку наступний: поточний робочий каталог, каталог, у якому знаходиться джерело, каталог MPASM.

Приклад:

include "c:\sys\sysdefs.inc" ; system defs

include <addmain.asm> ; register defs

LIST – установити параметри лістингу

Синтаксис:

list [<list_option>, , <list_option>]

Директива <list> дозволяє вивід лістингу, якщо він до цього був заборонений. Крім того, один з параметрів лістингу може бути змінений для управління процесом асемблювання відповідно до табл. 6.5.

Табл. 10.5. Параметри, які використовуються директивою list.

Параметр	**Значення за замовчуванням**	**Опис**
C=nnn	80	Кількість символів у рядку
n=nnn	59	Кількість рядків на сторінці
t=ON|OFF	OFF	Укорочувати рядки лістингу
p=<type>	None	Встановити тип процесора: PIC16C54, PIC16C84, PIC16F84, PIC17C42 та ін.
r=<radix>	HEX	Встановити систему числення за замовчуванням: hex, dec, oct.
w=<level>	0	Встановити рівень повідомлень діагностики у файлі лістингу: 0 - виводити усі повідомлення; 1 - виводити попередження і помилки; 2 - виводити тільки помилки.
x=ON|OFF	OFF	Включити чи виключити макророзширення.

NOLIST – виключити вихід лістингу

Синтаксис:

NOLIST

ORG – встановити початкову адресу програми

Синтаксис:

<label> org <expr>

Встановлює початкову адресу програми для наступного коду відповідно до адреси в <expr>. MPASM виводить переміщуваний об'єктний код, а MPLINK розмістить код за визначеною адресою. Якщо мітка <label> визначена, то їй буде присвоєне значення <expr>. За замовчуванням початкова адреса має нульове значення. Директива може не використовуватися, якщо створюється об'єктний модуль.

Приклад:

int_1 org 0x20; Перехід за вектором 20

int_2 org int_1+0x10; Перехід за вектором 30

PROCESSOR – установити тип процесора

Синтаксис:

processor <processor_type>

Установлює тип використовуваного процесора <processor_type>:

[16C54 | 16C55 | 16C56 | 16C57 | 16C71 | 16C84 | 16F84 | 17C42].

Загальні процесорні сімейства можуть бути обрані як:[16C5X | 16CXX | 17CXX]

Для підтримки сумісності з новими виробами вибирається максимум доступної пам'яті.

SET – визначити асемблерну змінну

Синтаксис:

<label> set <expr>

Директива SET функціонально еквівалентна директиві EQU, за винятком того, що величина, обумовлена SET, може бути змінена директивою SET.

Приклад:

area set 0

widthset 0x12

length set 0x14

area set length * width

length set length + 1

TITLE – визначити програмний заголовок

Синтаксис:

title "<title_text>"

Ця директива встановлює текст, що використовується у верхній лінії сторінки лістингу. <title_text> - це друкована ASCII послідовність, поміщена в подвійні дужки. Вона може бути до 60 символів довжиною.

Приклад:

title "operational code, rev 5.0"

10.2. Компоновник MPLINK

Абсолютний (непереміщуваний) код програми генерується безпосередньо при асемблюванні та розташовується в програмній пам'яті в порядку проходження операторів програми. Оператори переходу на мітку відразу ж заміняються відповідним кодом переходу на адресу мітки.

При генерації переміщуваного коду кожна секція програмного коду повинна випереджатися директивою CODE. Остаточне розміщення програмних кодів, розміщення фізичних адрес переходів виконує компоновник MPLINK.

Компоновник MPLINK виконує наступні задачі:

- розподіляє коди і дані, тобто визначає, у якій частині програмної пам'яті будуть розміщені коди й у яку область ОЗП будуть поміщені змінні;
- розподіляє адреси, тобто привласнює посиланням на зовнішні об'єкти в об'єктному файлі конкретні фізичні адреси;
- генерує код, що виконується, тобто видає файл у форматі .hex, що може бути записаний у пам'ять МК;
- відслідковує конфлікти адрес, тобто гарантує, що програма чи дані не будуть розміщатися в просторі адрес, що уже зайняті;
- надає символьну інформацію для налагодження.

Для більш докладного вивчення роботи компоновника варто звернутися до спеціальної літератури.

10.3. Менеджер бібліотек MPLIB

Менеджер бібліотек дозволяє створювати і модифікувати файли бібліотек. Бібліотечний файл є колекцією об'єктних модулів, які розміщені в одному файлі. MPLIB використовує об'єктні модулі з ім'ям типу "filename.o" формату COFF (Common Object File Format).

Використання бібліотечних файлів спрощує компонування програми, робить її більш структурованою і полегшує її модифікацію.

10.4. Симулятор MPSIM

Симулятор MPSIM це симулятор подій, призначений для налагодження програмного забезпечення PIC-контролерів. MPSIM моделює усі функції контролера, включаючи всі режими скидання, функції таймера/лічильника, роботу сторожового таймера, режими SLEEP і Power-down, роботу портів вводу/виводу.

MPSIM запускається з командного рядка DOS, конфігурується користувачем і безпосередньо застосовує вихідні дані асемблера MPASM.

Перед використанням симулятора необхідно відасемблювати вихідний файл <file_name>.asm і одержати файл об'єктного коду у форматі INHX8M, створюваний MPASM за замовчуванням:

MPASM <file_name>.asm <RETURN>

Щоб запустити симулятор, необхідно набрати в командному рядку:

MPSIM<RETURN>.

Вид екрана, одержуваного при запуску MPSIM, показаний на рис. 6.2. Екран розділений на три частини (вікна). У верхньому вікні показаний поточний стан моделювання, включаючи програму, що моделюється, тип МК, кількість виконаних командних циклів і витрачений на них час. Середнє вікно використовується для виводу вмісту регістрів користувача. Набір регістрів і формат виведених на екран даних визначаються файлом MPSIM.INI, що далі буде описаний докладніше. Нижнє вікно містить запрошення на введення команд, а також поточні операції і результат їхнього виконання.

При запуску симулятор MPSIM починає шукати командний файл MPSIM.INI. Цей текстовий файл створюється користувачем і використовується для задавання всіх задіяних у програмі параметрів.

```
User4 RADIX=X MPSIM 5.20 16c84 TIME=0.0u 0 ?=Help
W: 00 F1: 00 F2: 1FF F3: 0001111 IOA: 0F F5: 0F

%P84                ;Choose Microcontroller number = 84
%SR X               ;Set Input/Output radix to
heXadecimal
%ZR                 ;Set all registers to 0
%ZT                 ;Zero elapsed time counter to 0
%RE                 ;Reset elapsed time and step
count
%V W,X,2            ;register W
%AD  F1,X,2         ;register TMR0
%AD  F2,X,3         ;register PCL
%AD  F3,B,8         ;register STATUS
%AD  IOA,X,2        ;Port "A" TRIS register
%AD  F5,X,2         ;Port "A" register
%RS                 ;Reset
%SC 1               ;Set the clock 1MHz
%LO user4
Hex code loaded
Listing file        loaded
Symbol table        loaded
218960 bytes        memory free
%_
```

Рис. 10.2. Вид робочого вікна симулятора MPSIM.

Один із прикладів файлу MPSIM.INI наведений нижче:

```
                ; MPSIM file for user4
P84             ;використання МК сімейства PIC16C84
SR X            ;подання даних у шіснадцятковому форматі
ZR              ;скидання регістрів МК у нуль
ZT              ;скидання таймера в нуль
RE              ;скидання часу виконання команди і лічильника циклів
V W,X,2         ;вивід регістра W у hex форматі на два знакомісця
AD F1,X,2       ;вивід на екран регістра TMR0 у hex форматі на два знакомісця
AD F2,X,3       ;вивід на екран регістра PCL у hex форматі на три знакомісця
```

AD F3,B,8 ;вивід на екран регістра STATUS у bin форматі на вісьм знакомісць

AD IOA,X,2 ;вивід на екран регістра TRISA у hex форматі на два знакомісця

AD F5,X,2 ;вивід на екран регістра порту A у hex форматі на два знакомісця

SC 1 ;установка тактової частоти 1 МГц

RS ;ініціалізація МК

LO user4

У наведеному файлі зазначено: тип мікроконтролера, система числення даних за замовчуванням, регістри, уміст яких виводиться на екран, спосіб подання даних, робочі параметри. Будь-яка команда, яка виконується MPSIM, може бути задана у файлі MPSIM.INI, що визначає початковий стан програми. При роботі MPSIM створює файл MPSIM.JRN, у якому зберігаються всі відомості про натискання клавіш у процесі роботи.

У файлі MPSIM.INI допускається вводити коментарі, які даються після знаку ";", але не допускається використання порожніх рядків.

Основні команди, які застосовуються в симуляторі MPSIM, наведені в табл. 10.6. Коли ці команди вводяться в сеансі роботи з MPSIM, вони заносяться у файл MPSIM.JRN, що використовується при створенні розширеного файлу MPSIM.INI. Даний файл можна задіяти для виявлення помилок і забезпечення нормального виконання програми після виправлення коду.

Табл. 10.6. Основні команди симулятора MPSIM.

Команда	Параметр	Коментарі
AB	–	Переривання поточної сесії
AD	Reg[, Radix[, Digits]]	Вивід умісту регістра на екран у зазначеному форматі і заданій системі числення X, B чи D
B	[addr]	Установка крапки зупинки на поточній чи зазначеній адресі
C	[#break]	Продовження виконання програми з пропуском зазначеної кількості наступних крапок зупинки
DB	–	Вивід на екран всіх активних крапок зупинки
DI	[addr1[,addr2]]	Вивід на екран фрагмента пам'яті програм
DR	–	Вивід умісту всіх регістрів
DW	[E|D]	Дозвіл/заборона функціонування сторожового таймера
E	[addr]	Виконання програми з поточної чи зазначеної адреси
F	Reg	Вивід на екран умісту регістра і можливість його редагування користувачем

GE	filename	Одержання і виконання командного файлу. Це спосіб завантаження командного файлу .INI
GO	–	Запуск МК і початок виконання програми
IP	[time|step]	Введення вхідних сигналів у відповідності з значенням параметра step у файлі Stimulus
LO	filename	Завантаження в MPSIM файлів .HEX і .COD
M	addr	Вивід на екран умісту пам'яті програм, починаючи з адреси "addr" і можливість його редагування. Введення "Q" завершує команду.
P	device	Вибір типу модельованого МК
Q	–	Вихід з MPSIM і запис команд у файл .JRN
RE	–	Скидання часу виконання і лічильника циклів
RS	–	Скидання модельованого МК
SE	pin|port	Вивід на екран стану зазначеного виводу чи порту і можливість його зміни
SR	O|X|D	Установка системи числення за замовчуванням
SS	[addr]	Покрокове виконання, починаючи з зазначеної адреси. При відсутності адреси - виконання йде з поточного місця
ST	filename	Завантаження файлу стимуляції
W		Відображення стану регістра W з можливістю його модифікації
ZM	addr1,addr2	Очищення пам'яті програм з адреси addr1 по addr2
ZR	–	Скидання всіх регістрів МК
ZT	–	Скидання таймера/лічильника МК

Для моделювання зовнішніх тестових подій (сигналів) на модельований МК використовуються файли стимуляції з розширенням .STI. Ці файли використовуються MPSIM для того, щоб забезпечити подачу однократних і повторюваних вхідних сигналів у процесі виконання програми. При цьому можна спостерігати на екрані, як МК реагує на сигнали.

Як приклад нижче наведений файл для тестування програми, що виконує опитування стану лінії 1 порту А.

```
! test1.STI
STEP    RA1
   1     1 !Установка на вході RA1 стану "1"
 200     0 !Надходження на вхід RA1 сигналу "0"
1000     1 !Перехід сигналу на вході RA1 у "1"
1200     0 !Повторна подача нульового сигналу
```

Файл сигналу складається з безлічі станів, для яких задається параметр STEP, що визначає кількість циклів, протягом яких підтримується зазначений стан. Він дозволяє одночасно подавати сигнали на різні виводи МК. У файлі сигналу можна вказати будь-який вивід МК, у тому числі і вивід ініціалізації (_MCLR). Для позначення коментарів використовується знак !.

ТЕМА 11. ОСОБЛИВОСТІ ПРОЕКТУВАННЯ І ПРАКТИЧНОГО ЗАСТОСУВАННЯ МІКРОПРОПРО-ЦЕСОРНИХ ПРИСТРОЇВ ДЛЯ ПОБУТОВОЇ ТЕХНІКИ І СИСТЕМ КЕРУВАННЯ.

МПС на основі МК використовуються найчастіше як убудовані системи для рішення задач управління деяким об'єктом. Важливою особливістю даного застосування є робота в реальному часі, тобто забезпечення реакції на зовнішні події протягом визначеного часового інтервалу. Такі пристрої одержали назву контролерів.

Технологія проектування контролерів на базі МК цілком відповідає принципу нерозривного проектування і **відлагодження апаратних і програмних засобів**, прийнятому в мікропроцесорній техніці. Це означає, що перед виробником такого роду МПТ стоїть задача реалізації повного циклу проектування, починаючи від **розробки** алгоритму функціонування і закінчуючи комплексними іспитами в складі виробу, а, можливо, і супроводом при виробництві. Сформована до даного часу методологія проектування контролерів може бути подана так, як показано на Рис. 11.1.

У технічному завданні формулюються вимоги до контролера з погляду реалізації визначеної функції управління. Технічне завдання містить у собі набір вимог, які визначають, що користувач хоче від контролера і що розроблювальний прилад повинен робити. Технічне завдання може мати вид текстового опису, не звільненого в загальному випадку від внутрішніх протиріч.

На підставі вимог користувача складається функціональна специфікація, що визначає функції, виконувані контролером для користувача після завершення проектування, уточнюючи тим самим, наскільки пристрій відповідає пропонованим вимогам. Вона містить у собі описи форматів даних, як на вході, так і на виході, а також зовнішні умови, що керують діями контролера.

Функціональна специфікація і вимоги користувача є критеріями оцінки функціонування контролера після завершення проектування. Може знадобитися проведення декількох ітерацій, що включають обговорення вимог і функціональної специфікації з потенційними користувачами контролера, і відповідну корекцію вимог і специфікації. Вимоги до типу використовуваного МК формулюються на даному етапі найчастіше в неявному виді.

Етап розробки алгоритму управління є найбільш відповідальним, оскільки помилки даного етапу зазвичай виявляються тільки при іспитах закінченого виробу і приводять до потреби в дорогій переробці всього пристрою. Розробка алгоритму звичайно зводиться до вибору одного з декількох можливих варіантів алгоритмів, що відрізняються співвідношенням об'єму програмного забезпечення й апаратних засобів.

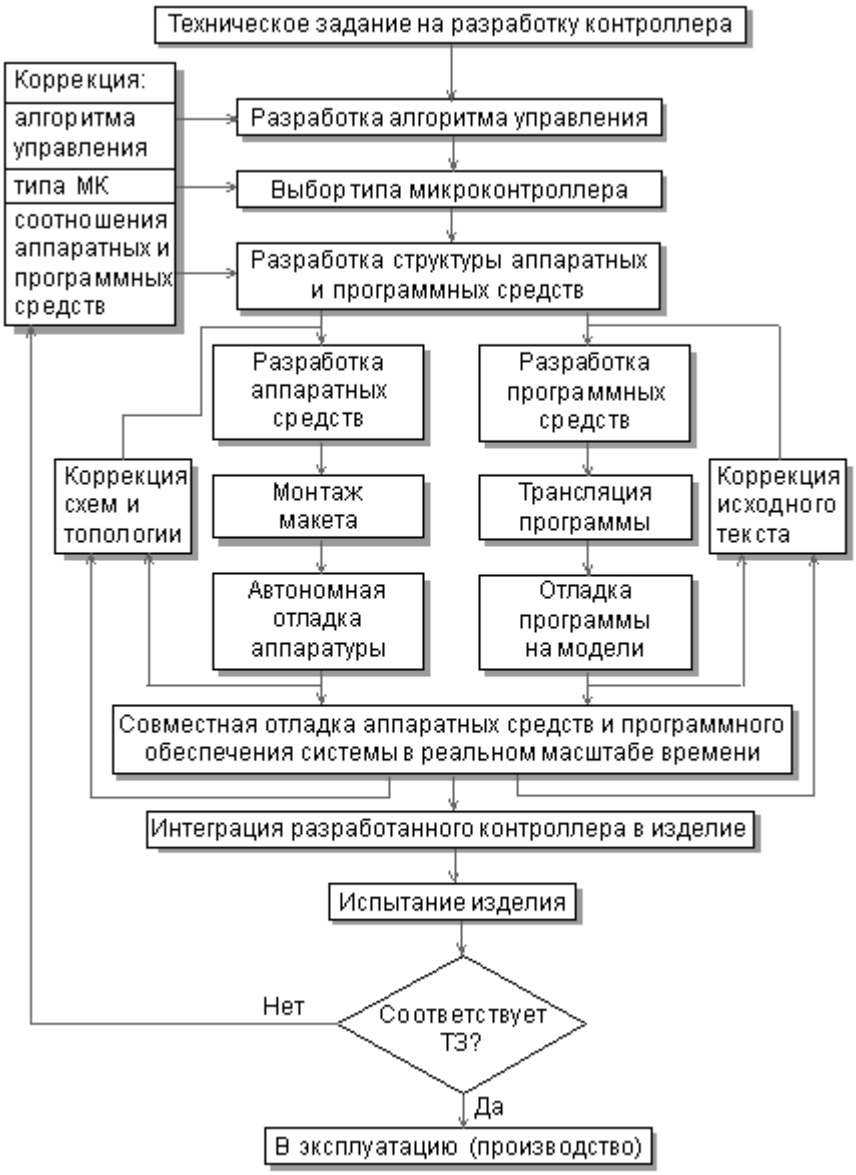


Рис. 11.1. Основні етапи розробки контролера.

При цьому необхідно виходити з того, що максимальне використання апаратних засобів спрощує розробку і забезпечує високу швидкодію контролера в цілому, але супроводжується, як правило, збільшенням вартості і споживаної потужності. Зв'язано це з тим, що збільшення частки апаратних засобів досягається або шляхом вибору більш складного МК, або шляхом використання спеціалізованих інтерфейсних схем. І те, і інше викликає зростання вартості й енергоспоживання. Збільшення питомої ваги програмного забезпечення дозволяє скоротити кількість елементів контролера і вартість апаратних засобів, але це приводить до зниження швидкодії, збільшенню необхідного об'єму внутрішньої пам'яті МК, збільшенню термінів розробки і налагодження програмного забезпечення. Критерієм вибору тут і далі є можливість максимальної реалізації заданих функцій програмними засобами при мінімальних апаратних витратах і за умови забезпечення заданих показників швидкодії і надійності в повному діапазоні умов експлуатації. Часто визначальними вимогами є можливість

захисту інформації (програмного коду) контролера, необхідність забезпечення максимальної тривалості роботи в автономному режимі й інші. У результаті виконання цього етапу остаточно формулюються вимоги до параметрів використовуваного МК.

При виборі типу МК враховуються наступні основні характеристики:

- розрядність;
- швидкодія;
- набір команд і способів адресації;
- вимоги до джерела живлення і споживана потужність у різних режимах;
- об'єм ПЗП програм і ОЗП даних;
- можливості розширення пам'яті програм і даних;
- наявність і можливості периферійних пристроїв, включаючи засоби підтримки роботи в реальному часі (таймери, процесори подій і т.п.);
- можливість перепрограмування в складі пристрою;
- наявність і надійність засобів захисту внутрішньої інформації;
- можливість постачання в різних варіантах конструктивного виконання;
- вартість у різних варіантах виконання;
- наявність повної документації;
- наявність і доступність ефективних засобів програмування і налагодження МК;
- кількість і доступність каналів постачання, можливість заміни виробами інших фірм.

Список цей не є вичерпним, оскільки специфіка проектованого пристрою може перенести акцент вимог на інші параметри МК. Визначальними можуть виявитися, наприклад, вимоги до точності внутрішнього компаратора чи напруг наявність великої кількості вихідних каналів ШІМ.

Номенклатура МК, які випускаються в даний час, визначається тисячами типів виробів різних фірм. Сучасна стратегія модульного проектування забезпечує споживача розмаїтістю моделей МК із тим самим процесорним ядром. Така структурна розмаїтість відкриває перед виробником можливість вибору оптимального МК, що не має функціональної надлишковості, що мінімізує вартість комплектуючих елементів.

Однак для реалізації на практиці можливості вибору оптимального МК необхідне досить глибоке пророблення алгоритму управління, оцінка об'єму програми, яка виконується, і кількості ліній з'єднання з об'єктом на етапі вибору МК. Допущені на даному етапі прорахунки можуть згодом привести до необхідності зміни моделі МК і повторного розведення друкованої плати макета контролера. У таких умовах доцільно виконувати попереднє моделювання основних елементів прикладної програми з використанням програмно-логічної моделі обраного МК.

При відсутності МК, який забезпечує необхідні згідно ТЗ характеристики проектованого контролера, необхідно повернутися до етапу розробки алгоритму управління і перегляд обраного співвідношення між об'ємом програмного забезпечення й апаратних засобів. Відсутність придатного МК

найчастіше означає, що для реалізації необхідного об'єму обчислень (алгоритмів управління) за відведений час потрібна додаткова апаратна підтримка. Негативний результат пошуку МК із необхідними характеристиками може бути зв'язаний також з необхідністю обслуговування великої кількості об'єктів управління. У цьому випадку можливе використання зовнішніх схем обрамлення МК.

На етапі розробки структури контролера остаточно визначається поєднання наявних і таких, які необхідно розробляти апаратних модулів, протоколи обміну між модулями, типи роз'ємів. Виконується попереднє пророблення конструкції контролера. У частині програмного забезпечення визначаються поєднання і зв'язки програмних модулів, мова програмування. На цьому ж етапі здійснюється вибір засобів проектування і налагодження.

Можливість перерозподілу функцій між апаратними і програмними засобами на даному етапі існує, але вона обмежена характеристиками вже обраного МК. При цьому необхідно мати на увазі, що сучасні МК випускаються, як правило, серіями (сімействами) контролерів, сумісних програмно і конструктивно, але, такі, що відрізняються за своїми можливостями (об'єм пам'яті, набір периферійних пристроїв і т.д.). Це дає можливість вибору структури контролера з метою пошуку найбільш оптимального варіанта реалізації.

Не можна не згадати тут про нову ідеологію розробки пристроїв на базі МК, що запропонована фірмою "Scenix". Вона заснована на використанні високошвидкісних RISC-мікроконтролерів серії SX з тактовою частотою до 100 Мгц. Ці МК мають мінімальний набір вмонтованої периферії, а усі більш складні периферійні модулі емулюються програмними засобами. Такі модулі програмного забезпечення називаються "віртуальними периферійними пристроями", вони забезпечують зменшення кількості елементів контролера, часу розробки, збільшують гнучкість виконання. До даного часу розроблені цілі бібліотеки віртуальних пристроїв, що містять налагоджені програмні модулі таких пристроїв як модулі ШІМ і ФАПЧ, послідовні інтерфейси, генератори і вимірювачі частоти, контролери переривань і багато чого іншого.

11.2. Розробка і налагодження апаратних засобів

Після розробки структури апаратних і програмних засобів подальша робота над контролером може бути розпаралелена. Розробка апаратних засобів містить у собі розробку загальної принципової схеми, розведення топології плат, монтаж макета і його автономне налагодження. Час виконання цих етапів залежить від наявного набору апробованих функціонально-топологічних модулів, досвіду і кваліфікації виробника. На етапі вводу принципової схеми і розробки топології використовуються, як правило, розповсюджені системи проектування типу "ACCEL EDA" чи "OrCad".

Автономне налагодження апаратури на основі МК із відкритою архітектурою припускають контроль стану багаторозрядних магістралей

адреси і даних з метою перевірки правильності звертання до зовнішніх ресурсів пам'яті і периферійних пристроїв. Закрита архітектура МК припускає реалізацію більшості функцій розроблювального пристрою внутрішніми засобами мікроконтролера. Тому розроблювальний контролер буде мати малу кількість периферійних ІС, а обмін з ними буде відбуватись переважно за послідовними інтерфейсами. Тут на перший план вийдуть питання узгодження за навантажувальною здатністю паралельних портів МК і налагодження алгоритмів обміну послідовними каналами.

11.3. Розробка і налагодження програмного забезпечення

Зміст етапів розробки програмного забезпечення, його трансляції і налагодження на моделях істотно залежить від використовуваних системних засобів. В даний час ресурси 8-розрядних МК достатні для підтримки програмування на мовах високого рівня. Це дозволяє використовувати всі переваги структурного програмування, розробляти програмне забезпечення з використанням роздільно трансльованих модулів. Одночасно продовжують широко використовуватися мови низького рівня типу асемблера, особливо при необхідності забезпечення контрольованих інтервалів часу. Задачі попередньої обробки даних часто вимагають використання обчислень із плаваючою комою, трансцендентних функцій.

В даний час самим потужним засобом розробки програмного забезпечення для МК є інтегровані середовища розробки, що мають у своєму складі менеджер проектів, текстовий редактор і симулятор, а також допускають підключення компіляторів мов високого рівня типу Паскаль чи Сі. При цьому необхідно мати на увазі, що архітектура багатьох 8-розрядних МК унаслідок малої кількості ресурсів, сторінкового розподілу пам'яті, незручної індексної адресації і деяких інших архітектурних обмежень не забезпечує компілятору можливості генерувати ефективний код. Для обходу цих обмежень виробники ряду компіляторів змушені були перекладати на користувача турботу про оптимізацію коду програми.

Для перевірки і налагодження програмного забезпечення використовуються так звані програмні симулятори, які надають користувачу можливість виконувати розроблену програму на програмно-логічній моделі МК. Програмні симулятори поширюються, як правило, безкоштовно і сконфігуровані відразу на кілька МК одного сімейства. Вибір конкретного типу МК серед моделей сімейства забезпечує відповідна опція меню конфігурації симулятора. При цьому моделюється робота ЦП, усіх портів вводу/виводу, переривань і іншої периферії. Карта пам'яті модельованого МК завантажується в симулятор автоматично, налагодження ведеться в символьних позначеннях регістрів.

Завантаживши програму в симулятор, користувач має можливість запускати її в покроковому чи безупинному режимах, задавати умовні чи безумовні точки зупинки, контролювати і вільно модифікувати вміст комірок пам'яті і регістрів симульованого МК.

11.4. Методи і засоби спільного налагодження апаратних і програмних засобів

Етап спільного налагодження апаратних і програмних засобів у реальному масштабі часу є самим трудомістким і вимагає використання інструментальних засобів налагодження. До числа основних інструментальних засобів налагодження

- відносяться:
- схемні емулятори;
- плати розвитку (оціночні плати);
- монітори налагодження;
- емулятори ПЗП.

Схемний емулятор - програмно-апаратний засіб, здатний замінити емульований МК у реальній схемі. Стикування схемного емулятора з відлагоджуваною системою відбувається за допомогою кабелю з спеціальною емуляційною голівкою, яка вставляється замість МК у відлагоджувану систему. Якщо МК не можна вийняти з відлагоджуваної системи, то використання емулятора можливе, тільки, якщо цей мікроконтролер має відлагоджувальний режим, при якому всі його виводи знаходяться в третьому стані. У цьому випадку для підключення емулятора використовують спеціальний адаптер-кліпсу, яка під'єднується безпосередньо до виводів емульованого МК.

Схемний емулятор - це найбільш потужний й універсальний відлагоджувальний засіб, який робить процес функціонування відлагоджуваного контролера прозорим, тобто легко контрольованим, довільно керованим і таким, що легко модифікується.

Плати розвитку, чи, як прийнято їх називати в закордонній літературі, оціночні плати (Evaluation Boards), є свого роду конструкторами для макетування електронних пристроїв. Звичайно це друкована плата з установленим на ній МК і всієї необхідної йому стандартної периферії. На цій платі також установлюють схеми зв'язку з зовнішнім комп'ютером. Як правило, там же є вільне поле для монтажу прикладних схем користувача. Іноді передбачено вже готове розведення для установки додаткових пристроїв, що рекомендуються фірмою. Наприклад, ПЗП, ОЗП, РКІ-дисплей, клавіатура, АЦП і ін. Крім навчальної чи макетної мети, такі дороблені користувачем плати можна використовувати як одноплатні контролери, які вбудовуються в малосерійну продукцію.

Для більшої зручності плати розвитку комплектуються ще і найпростішим засобом налагодження на базі **монітора налагодження**. Використовуються два типи моніторів налагодження: один для МК, що мають зовнішню шину, а другий - для МК, що не мають зовнішньої шини.

У першому випадку відлагоджувальний монітор поставляється у виді мікросхеми ПЗП, що вставляється в спеціальну розетку на платі розвитку. Плата також має ОЗП для програм користувача і канал зв'язку з зовнішнім комп'ютером чи терміналом. В другому випадку плата розвитку має вмонтовані схеми програмування внутрішнього ПЗП МК, які керуються від

зовнішнього комп'ютера. При цьому програма монітора просто заноситься в ПЗП МК разом із прикладними кодами користувача. Прикладна програма повинна бути спеціально підготовлена: у потрібні місця необхідно уставити виклики відлагоджцвальних підпрограм монітора. Потім здійснюється пробний прогін. Щоб внести в програму виправлення, користувачу треба стерти ПЗП і зробити повторний запис. Готову прикладну програму одержують з налагодженої шляхом видалення усіх викликів моніторних функцій і самого монітора налагодження. Можливості налагодження, надані комплектом "плата розвитку плюс монітор", не настільки універсальні, як можливості схемного емулятора, та й деяка частина ресурсів МК у процесі налагодження відбирається для роботи монітора. Проте, наявність набору готових програмно-апаратних засобів, що дозволяють без утрати часу приступити до монтажу і налагодження проектованої системи, у багатьох випадках є вирішальним чинником. Особливо якщо врахувати, що вартість такого комплекту трохи менша, ніж вартість більш універсального емулятора.

Емулятор ПЗП - програмно-апаратний засіб, що дозволяє заміщати ПЗП на відлагоджуваній платі, і підставляє замість нього ОЗП, у яке може бути завантажена програма з комп'ютера через один зі стандартних каналів зв'язку. Цей пристрій дозволяє користувачу уникнути багаторазових циклів перепрограмування ПЗП. Емулятор ПЗП потрібен тільки для МК, які можуть звертатися до зовнішньої пам'яті програм. Цей пристрій за складністю і вартістю можна порівняти з платами розвитку і має одну велику перевагу: універсальність. Емулятор ПЗП може працювати з будь-якими типами МК.

Емульована пам'ять доступна для перегляду і модифікації, але контроль над внутрішніми керуючими регістрами МК був донедавна неможливий.

Останнім часом з'явилися моделі інтелектуальних емуляторів ПЗП, які дозволяють "заглядати" усередину МК на платі користувача. Інтелектуальні емулятори це гібрид зі звичайного емулятора ПЗП, монітора налагодження і схем швидкого перемикання шини з емулятора на монітор. Це створює ефект, як якби монітор налагодження був установлений на платі користувача і при цьому він не займає в МК ніяких апаратних ресурсів, крім невеликої зони програмних кроків, приблизно 4К.

Етап спільного налагодження апаратних і програмних засобів у реальному масштабі часу завершується, коли апаратура і програмне забезпечення спільно забезпечують виконання всіх кроків алгоритму роботи системи. Наприкінці етапу налагодження програма заноситься за допомогою програматора в енергонезалежну пам'ять МК, і перевіряється робота контролера без емулятора. При цьому використовуються лабораторні джерела живлення. Частина зовнішніх джерел сигналів може моделюватися.

Етап інтеграції розробленого контролера у виріб полягає в повторенні робіт зі спільного налагодження апаратури і керуючої програми, але при роботі в складі виробу, живленні від штатного джерела і з інформацією від штатних джерел сигналів і датчиків.

Поєднання й об'єм випробувань розробленого і виготовленого контролера залежить від умов його експлуатації і визначається відповідними нормативними документами. Проведення випробувань таких функціонально складних виробів, як сучасні контролери, може вимагати розробки спеціалізованих засобів контролю стану виробу під час випробувань.

ТЕМА 12. ОЦІНКА МОЖЛИВОСТЕЙ І ОБГРУНТУВАННЯ ВИБОРУ АРХІТЕКТУРИ МІКРОПРОЦЕСОРНИХ ПРИСТРОЇВ ПРИ РОЗРОБЦІ ПОБУТОВОЇ ТЕХНІКИ.

Процес взаємодії людини з ЕОМ налічує вже з більш40лет. Донедавна у цьому могли брати участь тільки фахівці - інженери, математики - програмісти, оператори. Останніми роками сталися кардинальні зміни у області обчислювальної техніки. Завдяки розробки і впровадження мікропроцесорів до структури ЕОМ з'явилися малогабаритні, зручні для користувача персональні комп'ютери. Ситуація, у ролі користувача може бути лише фахівець із обчислювальної техніки, а й будь-якої людина, чи це школяр чи домогосподарка, лікар чи вчитель, робочий чи інженер. Часто це явище називають феноменом самого персонального комп'ютера. Нині світової парк персональних комп'ютерів перевищує 20 млн.

Чому виник цього прикрого феномена? Відповідь це питання можна знайти, якщо чітко сформулювати, що таке персонального комп'ютера та її це основна прикмета. Треба правильно сприймати сам означник "персональний", він означає приналежність комп'ютера людині на правах особистої власності. Визначення "персональний" виникло бо людина отримав таку можливість спілкуватися із ЕОМ без посередництва професіонала програміста, самостійно, персонально. У цьому необов'язково знати спеціальну мову ЕОМ. Існуючі в комп'ютері програмні кошти забезпечать сприятливу " дружню" форму діалогу користувача і ЕОМ.

Нині серед найпопулярніших комп'ютерів стали модель IBM PC і його модернізований варіант IBM PCXT, котрий за архітектурі, програмному забезпеченню, зовнішньому оформленню вважається базової моделлю самого персонального комп'ютера.

Основою самого персонального комп'ютера є системний блок. Він організує роботу, обробляє інформацію, виробляє розрахунки, забезпечує зв'язок чоловіки й ЕОМ. Користувач зобов'язаний досконально розумітися на тому, як працює системний блок. Це справа фахівців. Але повинен знати, з яких функціональних блоків полягає комп'ютер. Не маємо чіткого ставлення до принципі дії внутрішніх функціональних блоків навколишніх предметів - холодильника, газової плити, пральної машини, автомобіля, але мають знати, що основою роботи цих пристроїв, які можливості складових блоків.

12.1 Базова структура мікропроцесорної системи

Завдання управління системою доручається центральний процесор (ЦП), який із пам'яттю і політичною системою виводу-введення-висновку через канали пам'яті і вводу-виводу відповідно. ЦП зчитує з пам'яті команди, що утворюють програму і декодує їх. Відповідно до результатом декодування команд вона здійснює вибірку даних із пам'яті портів введення, опрацьовує і пересилає знову на пам'ять чи порти виведення. Існує й можливість виводу-введення-висновку даних із пам'яті на зовнішні пристрої і

назад, минаючи ЦП. Цей механізм називається прямим доступом до пам'яті (ПДП).

З погляду користувача під час виборів мікропроцесора доцільно розташовувати деякими узагальненими комплексними характеристиками можливостей мікропроцесора. Розробник потребує з'ясуванні і розумінні тільки для тих компонентів мікропроцесора, які позначаються на програмах повинні бути враховані розробки схем і програм функціонування системи. Такі характеристики визначаються поняттям архітектури мікропроцесора.

1.2 Поняття архітектури мікропроцесора

Архітектура типовою невеличкий обчислювальної системи з урахуванням мікро-ЕОМ показано на рис. 1. Така мікро-ЕОМ містить все 5 основних блоків цифровий машини: пристрій введення інформації, котра управляє пристрій (УУ), арифметико-логічного приструю (АЛУ) (що входять до склад мікропроцесора), запам'ятовуючі пристрої (ЗУ) і пристрій виведення інформації.

Мікропроцесор координує роботу всіх пристроїв цифровий системи з допомогою шини управління (ШУ). Крім ШУ є16-разрядная адресна шина (ША), яка служить для вибору певної осередки пам'яті, порту введення чи порту виведення. По 8-разрядной інформаційної шині чи шині даних (ШД) здійснюється двунаправленная пересилання даних до мікропроцесору і південь від мікропроцесора. Важливо, що МП може посилати інформацію на згадку про мікро-ЕОМ або до одного з портів виведення, і навіть отримувати з пам'яті чи то з однієї з портів введення.

Постійне запам'ятовуючий пристрій (ПЗУ) в мікро-ЕОМ містить деяку програму (практично програму ініціалізації ЕОМ). Програми може бути завантажені в запам'ятовуючий пристрій із довільною вибіркою (ЗУПВ) і із зовнішнього запомінаючого устрою (ВЗУ). Це програми користувача.

Як приклад, ілюструючого роботу мікро-ЕОМ, розглянемо процедуру, для реалізації якій теж потрібне виконати таку послідовність елементарних операцій:

1. Натиснути клавішу букві "А" на клавіатурі.
2.Поместить букву "А" на згадку про мікро-ЕОМ.
3. Вивести букву "А" на екран дисплея.

Це типова процедура ввода-запоминания-вивода, розгляд якого надає можливість пояснити принципи використання деяких пристроїв, які входять у мікро-ЕОМ.

Збережена програма містить такий ланцюжок команд:

1. Запровадити дані з порту введення 1.
2. Запам'ятати дані в осередку пам'яті 200.
3.Переслать дані до порту виведення 10.

У цьому програмі всього три команди, хоча рис. 2 може бути, що у пам'яті програм записано шість команд. Це з тим, що команда зазвичай розбивається на частини. Перша частина команди 1 в наведеної вище програмі - команда введення даних. В другій частині команди 1 вказується,

звідки слід впровадити дані (з порту 1). Перша частина команди, відповідно до якої конкретне дію, називається кодом операції (КОП), а друга частина - операндом. Код операції, і операнд розміщуються окремими осередках пам'яті програм. На рис. 2 КОП зберігається в осередку 100, а код операнда - в осередку 101 (порт 1); останній вказує звідки треба взяти інформацію.

У МП на рис. 2 виділено решта 2 нових блоку - регістри: акумулятор реєстр команд.

Розглянемо проходження команд та об'єктивності даних всередині мікро-ЕОМ з допомогою занумерованих гуртків з діаграми. Нагадаємо, що мікропроцесор - це центральний вузол, управляючий переміщенням всіх даних, і виконанням операцій.

Отже, і під час типовою процедури ввода-запоминания-вивода в мікро-ЕОМ відбувається наступна послідовність дій:

1. МП видає адресу 100 на шину адреси. По шині управління надходить сигнал, який встановлює пам'ять програм (конкретну мікросхему) в режим зчитування.

2. ЗУ програм пересилає першу команду ("Запровадити дані") по шині даних, і МП отримує це закодоване повідомлення. Команда міститься у регістр команд. МП декодує (інтерпретує) отриману команду яких і визначає, що з команди потрібен операнд.

3. МП видає адресу 101 на ША; ШУ використовується для перекладу пам'яті програм, у режим зчитування.

4. З пам'яті програм на ШД пересилається операнд "З порту 1". Цей операнд перебуває у програмної пам'яті в осередку 101. Код операнда (у якому адресу порту 1) передається по ШД до МП і направляється в регістр команд. МП тепер декодує повну команду ("Запровадити дані з порту 1").

5. МП, використовуючи ША і ШУ, пов'язані з побудовою введення, відкриває порт 1. Цифровим код літери "А" передається в акумулятор всередині МП і запоминается. Важно відзначити, що з обробці кожної програмної команди МП діє відповідно до микропроцедуре виборки-декодирования-исполнения.

6. МП звертається до осередку 102 по ША. ШУ використовується для перекладу пам'яті програм, у режим зчитування.

7. Код команди "Запам'ятати дані" подається на ШД і пересилається в МП, де міститься у регістр команд.

8. МП дешифрирует цю команду яких і визначає, що з неї потрібен операнд. МП звертається до осередку пам'яті 103 і приводить у активний стан вхід зчитування мікросхем пам'яті програм.

9. З пам'яті програм на ШД пересилається код повідомлення "У осередку пам'яті 200". МП сприймає цей операнд й поміщає їх у регістр команд. Повна команда "Запам'ятати дані в осередку пам'яті 200" обрано з пам'яті програм, тождекодирована.

10. Тепер починається процес виконання команди. МП пересилає адресу 200 на ША і активізує вхід записи, належить до пам'яті даних.

11. МП спрямовує що зберігається в акумуляторі інформацію на згадку про даних. Код літери "А" передається по ШД і записується в осередок 200 цієї пам'яті. Виполнена друга команда. Процес запам'ятовування не руйнує вмісту акумулятора. У ньому продовжує перебувати код літери "А".

12. МП звертається до осередку пам'яті 104 для вибору черговий команди, і переводить пам'ять програм, у режим зчитування.

13. Код команди виведення даних пересилається по ШД до МП, який поміщає їх у регістр команд, дешифрирует яких і визначає, що потрібен операнд.

14. МП видає адресу 105 на ША й встановлює пам'ять програм, у режим зчитування.

15. З пам'яті програм по ШД до МП надходить код операнда "У порт 10", який далі міститься у регістр команд.

16. МП дешифрирует повну команду "Вивести дані до порту 10". З допомогою ША і ШУ, що пов'язують його з побудовою виведення, МП відкриває порт 10, пересилає код літери "А" (досі що у акумуляторі) по ШД. Буква "А" виводиться через порт 10 на екран дисплея.

У багатьох мікропроцесорних систем (МШС) передача інформації здійснюється способом, аналогічним розглянутому вище. Найбільш істотні розбіжності можливі в блоках введення та виведення інформації.

Підкреслимо вкотре, що став саме мікропроцесор є ядром системи та здійснює керівництво всіма операціями. Його робота представляє послідовну реалізацію мікропроцедур виборки-дешифрации-исполнения. Проте фактична послідовність операцій на МШС визначається командами, записаними у пам'яті програм.

Отже, в МШС мікропроцесор виконує такі функції:

- вибірку команд програми з основний пам'яті;

-дешифрацію команд;

- виконання арифметичних, логічних та інших операцій, закодованих в командах;

- управління пересилкою інформації між регістрами і основний пам'яттю, між пристроями ввода/вивода;

- відпрацювання сигналів від пристроїв ввода/вивода, зокрема реалізацію переривань з цих пристроїв;

- управління економіки й координацію роботи основних вузлів МП.

1.3 Огляд існуючих типів архітектури мікропроцесорів

Є кілька підходів до класифікації мікропроцесорів на кшталт архітектури. Так, виділяють МП з CISC (Complete Instruction Set Computer) архітектурою, що характеризується повний набір команд, iRISC (Reduce Instruction Set Computer) архітектурою, що визначає систему зі скороченою набором команд однакового формату, виконуваних за такт МП.

Визначаючи як основного характеристики МП розрядність, виділяють такі типи МП архітектури:

- з фіксованою розрядністю і списком команд (однокристальние);

- з нарощуваної розрядністю (секційні) і мікропрограмним управлінням.

Аналізуючи адресні простору програм, тож даних, визначають МП з архітектурою фон Неймана (пам'ять програм, тож пам'ять даних перебувають у єдиний простір немає жодних ознак, вказують на той тип інформацією осередку пам'яті) і МП з архітектурою Гарвардської лабораторії (пам'ять програм, тож пам'ять даних розділені, мають адресні простору й способи доступу до них).

Розглянемо докладніше основні типи архітектурних рішень, виділяючи зв'язок із способами адресації пам'яті.

1.Регистровая архітектура визначається наявністю досить великої регістрового файла всередині МП. Команди отримують унікальну можливість звернутися до операндам, розміщеним у одній з двох запам'ятовувальних середовищ: оперативної пам'яті чи регістрах. Розмір регістру зазвичай фіксований і збігаються з розміром слова, фізично реалізованого оперативному пам'яті. До будь-якого регістру можна безпосередньо, оскільки регістри представлені у вигляді масиву запам'ятовувальних елементів регістрового файлу. Типовим є виконання арифметичних операцій лише у регістрі, у своїй команда містить два операнда (обидва операнда в регістрі чи один операнд в регістрі, а другий у оперативної пам'яті).

До даному типу архітектури належить мікропроцесор фірми Zilog. Процесор Z80 - дітище фірми Zilog крім розширеній системи команд, одного номіналу харчування й уміння виконувати програми, написані для i8080, мав архітектурні "родзинки".

На додачу до основного набору РОН, в кристалі реалізували другий комплект аналогічних регістрів. Це значно спрощувало роботу при виклик підпрограм чи процедур обслуговування переривань, оскільки програміст міг використовуватиме них альтернативний набір регістрів, уникаючи збереження в стеці вмісту РОНов для програми з допомогою операцій PUSH. З іншого боку, до системи команд було включено ряд спеціальних інструкцій, орієнтованих обробку окремих бітів, а підтримки регенерації динамічної пам'яті в схему процесора запроваджені відповідні апаратні кошти.Z80 застосовувався у машинах Sinclair ZX, Sinclair Spectrum, Tandy TRS80.

Граничний варіант - архітектура з адресацією у вигляді акумуляторів (менший набір команд).

МП фірми Motorola мав низку істотних переваг. Насамперед, кристалМС6800 вимагав до роботи одного номіналу харчування , а система команд була дуже прозорою для програміста. Архітектура МП також мала ряд особливостей.

Мікропроцесор МС 6800 містив два акумулятора, і результати операції АЛУ міг стати поміщений у кожної. Але найціннішою якістю структури МС 6800 було автоматичне збереження в стеці вмісту всіх регістрів процесора

при обробці переривань (>Z80 вимагалося при цьому кілька команд PUSH). Процедура відновлення РОН з стека теж виконувалася апаратно.

2.Стековая архітектура дає створити полі пам'яті з упорядкованою послідовністю запису і вибірки інформації.

У випадку команди неявно адресуються до елементу стека, розташованому з його вершині, або до двом верхнім елементам стека.

3. Архітектура МП, орієнтована на оперативну пам'ять (типу "память-память"), забезпечує високу швидкість праці та велику інформаційну ємність робочих регістрів і стека за її організації у оперативної пам'яті.

Архітектура цього передбачає явного визначення акумулятора, регістрів загального призначення чи стека; все операнди команд адресуються до області основний пам'яті.

З погляду важливості для пользователя-програміста під архітектурою у випадку розуміють сукупність наступних компонентів і характеристик:

- розрядності адрес і передачею даних;
- складу, імен та призначення програмно-доступних регістрів;
- форматів і системи команд;
- режимів адресації пам'яті;
- способів машинного уявлення даних різного типу;
- структури адресного простору;
- способу адресації зовнішніх пристроїв і коштів операцій ввода/вивода;
- класів переривань, особливостей ініціювання і методи обробки переривань.

12.2. Пристрій управління

Коди операції команд програми, надаються до сприймання керуючої частиною мікропроцесора, розшифровані і перетворені на ній, дають інформацію у тому, які операції треба виконати, де у пам'яті розташовані дані, куди треба спрямовувати результат і розташована наступна за виконуваної команда.

Керуючий пристрій має досить коштів у тому, щоб після сприйняття й інтерпретації інформації, яку за команді, забезпечити переключення (спрацьовування) всіх необхідних функціональних частин машини, і навіть у тому, щоб підвести до них дані і сприйняти отримані результати. Саме спрацьовування, т. е. зміна стану двійкових логічних елементів на протилежне, дозволяє у вигляді комутації вентилів виконувати елементарні логічні і арифметичні дії, і навіть передавати необхідні операнди в функціональні частини мікро-ЕОМ.

Пристрій управління у суворої послідовності у межах тактових і циклових тимчасових інтервалів роботи мікропроцесора (такт - мінімальний робочий інтервал, протягом якого відбувається одне елементарне дію; цикл - інтервал часу, протягом якого виконується одна машинна операція) здійснює: вибірку команди; інтерпретацію її з єдиною метою аналізу формату, службових ознак і обчислення адреси операнда (операндів); встановлення

такий і тимчасової послідовності всіх функціональних управляючих сигналів; генерацію управляючих імпульсів і передачу на управляючі шини функціональних частин мікро-ЕОМ і вентилі з-поміж них; аналіз результату операції, і зміна свого майна те щоб визначити місце розташування (адресу) наступній команди.

12.3. Особливості програмного імикропрограммного управління

У мікропроцесорах використовують два методу вироблення сукупності функціональних управляючих сигналів: програмний і мікропрограмний.

Виконання операцій на машині зводиться до елементарним перетворенням інформації (передача інформації між вузлами в блоках, зрушення інформацією вузлах, логічні поразрядні операції, перевірка умов тощо.) в логічних елементах, вузлах і блоках під впливом функціональних управляючих сигналів блоків (пристроїв) управління. Елементарні перетворення,не разложимие більш прості, виконуються протягом такту сигналів синхронізації і називаються мікрооперацiями.

У апаратних (схемних) пристроях управління кожну операцію відповідає свій набір логічних схем, які б виробляли певні функціональні сигнали до виконання мікрооперацій у визначені моменти часу. У цьому способі побудови устрою управління реалізація мікрооперацій досягається з допомогою якось з'єднаних між собою логічних схем, тому ЕОМ з апаратним пристроєм управління називають ЕОМ з жорсткою логікою управління. Це належить до фіксації системи команд у структурі зв'язків ЕОМ і означає практичну неможливість яких-небудь змін до системі команд ЕОМ після його виготовлення.

При мікропрограмній реалізації устрою управління у склад останнього вводиться ЗУ, кожен розряд вихідного коду якого визначає поява певного функціонального сигналу управління. Тому кожного мікрооперації ставлять у відповідність свій інформаційний код - мікрокоманда. Набір мікрокоманд і послідовність реалізації забезпечують виконання будь-який складної операції. Набір мікрооперацій називають мікропрограмами. Спосіб управління операціями шляхом послідовного зчитування і інтерпретації мікрокоманд з ЗУ (найчастіше як мікропрограмного ЗУ використовують швидкодіючі програміруємі логічні матриці), і навіть використання кодів мікрокоманд для генерації функціональних управляючих сигналів називають мікропрограмним, а мікро-ЕОМ з такою способом управління - мікропрограмнимі чи з береженої (гнучкою) логікою управління.

До мікропрограмам пред'являють вимоги функціональної повноти і мінімальності. Першу вимогу необхідне забезпечення можливості розробки мікропрограм будь-яких машинних операцій, а друге пов'язана з бажанням зменшити обсяг використовуваного устаткування. Облік чинника швидкодії веде до розширення мікропрограм, оскільки ускладнення останніх дозволяє скоротити час виконання команд програми.

Перетворення інформації виконується в універсальному аріфметико-логічном блоці мікропроцесора. Він зазвичай будується з урахуванням комбінаційних логічних схем.

Для прискорення виконання зазначених операцій вводяться додатково спеціальні операційні вузли (наприклад, циклічні сдвіги). З іншого боку, у складі мікропроцесорного комплекту (МПК) БІС вводяться спеціалізовані оперативні блоки арифметичних расширителей.

Операційні можливості мікропроцесора можна розширити рахунок збільшення числа регістрів. Якщо регістровом буфері закріплення функцій регістрів відсутня, їх можна використовувати як зберігання даних, так зберігання адрес. Такі регістри мікропроцесора називаються регістрами загального призначення (РОН). З розвитком технології реально здійснено виготовлення в мікропроцесорі 16, 32 і більше регістрів.

А загалом, принцип мікропрограмного управління (ПМУ) входять такі позиції:

1) будь-яка операція, реалізована пристроєм, є послідовністю елементарних дій мікрооперацій;

2) керувати порядком прямування мікрооперацій використовуються логічні умови;

3) процес операцій у пристрої описується у вигляді алгоритму, подається в термінах мікрооперацій і логічних умов, званого мікропрограмою;

4) мікропрограма використовують як форма уявлення функції устрою, з урахуванням якої визначаються структура і Порядок функціонування устрою у часі.

ПМУ забезпечує гнучкість мікропроцесорної системи та дозволяє здійснювати проблемну орієнтацію мікро- і міні-ЕОМ.

12.4. Режими адресації

Для взаємодії з різними модулями в ЕОМ би мало бути кошти ідентифікації осередків зовнішньої пам'яті, осередків внутрішньої пам'яті, регістрів МП і регістрів пристрої вввода/вивода. Тому кожного з запам'ятовувальних осередків присвоюється адресу, тобто. однозначна комбінація біт. Кількість біт визначає число ідентифікованих осередків. Зазвичай ЕОМ має різні адресні простору пам'яті і регістрів МП, інколи ж - окремі адресні простору регістрів пристрої вввода/вивода і внутрішньої пам'яті. З іншого боку, пам'ять зберігає як дані, і команди. Тож ЕОМ розроблено безліч способів звернення до пам'яті, званих режимами адресації.

Режим адресації пам'яті - це процедура чи схема перетворення адресної інформацію про операнда у його виконавчий адресу.

Усі способи адресації пам'яті можна розділити на:

1) прямий, коли виконавчий адресу береться безпосередньо з команди чи обчислюється з допомогою значення, вказаної у команді, і вмісту будь-якого регістру (пряма адресація, реєстрова, базова, індексна тощо.);

2) непрямий, який передбачає, що у команді міститься значення непрямого адреси, тобто. адреси осередки пам'яті, у якій перебуває остаточний виконавчий адресу (непряма адресація).

У кожній мікро-ЕОМ реалізовані лише окремі режими адресації, використання є, зазвичай, визначається архітектурою МП.

Укладання

Кількість персональних комп'ютерів як і світі, і, зокрема, у Росії різко зростає; ринок ПК – найперспективніший і дохідний серед інших ринків обчислювальної техніки. У північної Америці й Західної Європи відсоток сімей, мають ПК, наближається до 30. Безперечно, в наші дні кожен має вивчити і зрозуміти комп'ютер як теоретично, але, що є, та практично.

Аналіз нових рішень побудови структури комп'ютера показує, що процесор, пам'ять, устрою введення - виведення становлять основу будь-якого комп'ютера. Розглянемо найбільш поширену структурну схему, що лежить основу найчастіше трапляються моделей комп'ютерів, зокрема персональних.

Сучасний комп'ютер можна здебільшого спрощеної структурної схемою, де виділено центральна і периферійна частини. До центральній частині ставляться процесор й внутрішня соціальність пам'ять, до периферійної частини - устрою вводу-виводу й зовнішня пам'ять. У основу спрощеної структурної схеми закладено принципи магістральності, модульности, мікропрограмирумостью.

Не слід сподіватися, що успішний розвиток обчислювальної техніки якось кардинально переверне існування. Комп'ютер трохи більше (але й менш) ніж одне із потужних двигунів прогресу (як енергетика, металургія, хімія, машинобудування), який утруднює свої "залізні плечі" цю важливу функцію, як рутину обробки інформації. Ця рутина завжди і скрізь супроводжує найвищі польоти людську думку. Саме в рутині часто-густо тонуть зухвалі рішення, недоступні комп'ютера. Тому вкрай важливо " звалити" на комп'ютер рутинні операції, щоб звільнити людини щодо його істинного призначення творчества.

Майбутнє мікропроцесорної техніки пов'язано сьогодні з цими двома новими напрямами - нанотехнологіями і квантовими обчислювальними системами. Ці поки що переважно теоретичні дослідження стосуються використання газу як компонентів логічних схем молекул і навіть субатомних частинок: підвалинами обчислень повинні служити не електричні ланцюга, як тепер, а становище окремих атомів чи напрямок обертання електронів. Якщо "мікроскопічні" комп'ютери буде створено, всі вони обминуть сучасні машини багатьма суттєвими параметрами.