Міністерство освіти і науки України

Національний університет водного господарства та природокористування

С.Ю.Бочаров

Мікропроцесорна техніка

Навчальний посібник

Рекомендовано Міністерством освіти і науки України як навчальний посібник для студентів вищих навчальних закладів

Рівне – 2006

УДК 681.3.004.382.7 (075.8)
ББК 32.973я 7
Б 86

Гриф надано Міністерством освіти і науки України
(Лист №14/18.2-1564 від 04.07.05)

Рецензенти: Г.О. Козлик, доктор техн. наук, професор, заслужений діяч науки і техніки України, заступник Генерального директора НВК „Київський інститут автоматики”;
М.Д. Гераїмчук, доктор техн. наук, професор, завідувач кафедри приладобудування Приладобудівного факультету Національного технічного університету України „Київський політехнічний інститут”;
Д.В. Корбутяк, доктор фіз.-мат. наук, професор, завідувач відділу Інституту фізики напівпровідників НАН України.

Бочаров С.Ю. Мікропроцесорна техніка: Навчальний посібник – Рівне : 2006. - 163 с.
ISBN

У навчальному посібнику розглянуто класифікація та напрямки розвитку мікропроцесорної техніки, системи числення і дії з двійковими числами, логіка цифрових пристроїв, їх основні елементи і вузли. Описано архітектуру і модулі мікропроцесорних керуючих пристроїв – мікроконтролерів. Викладено основи програмування мікропроцесорних систем мовою асемблера та принципи комп’ютеризації вузлів і агрегатів машин. З метою кращого засвоєння матеріалу всі розділи посібника проілюстровано прикладами.

Видання призначене для студентів вищих навчальних закладів, що навчаються за професійним спрямуванням „Підйомно-транспортні, будівельні, дорожні, меліоративні машини і обладнання”, „Автомобілі і автомобільне господарство”, і може використовуватись студентами спеціальності „Автоматизоване управління технологічними процесами”.

УДК 681.3.004.382.7 (075.8)
ББК 32.973я 7

ISBN

Зміст

Передмова

Навчальний посібник написано для студентів напряму підготовки „Інженерна механіка” за спеціальностями „Підйомно-транспортні, будівельні, дорожні, меліоративні машини і обладнання” , „Автомобілі і автомобільне господарство”. Він може використовуватись студентами спеціальності „Автоматизоване управління технологічними процесами”.

Метою вивчення дисципліни є засвоєння принципів побудови, функціонування та програмування мікропроцесорних систем для автоматичного контролю та керування. В розділах навчального посібника розглянуто такі основні питання: класифікація, стан і перспективи розвитку сучасних мікропроцесорних керуючих пристроїв – мікроконтролерів; системи числення і дії з двійковими числами; основи логіки мікропроцесорних систем; елементи і вузли мікропроцесорних систем; їх архітектура та модульні пристрої; програмування мікропроцесорних систем та види програм. Також розглянуто мікропроцесорні системи керування агрегатами машин. З метою кращого засвоєння всі розділи посібника містять приклади. В результаті вивчення матеріалу, наведеного у посібнику, студенти набувають навики аналізу та використання сучасної мікросхемотехніки і мікропроцесорних систем для автоматизованого управління технологічними процесами.

Вступ

Курс „Мікропроцесорна техніка" вивчає основи побудови мікропроцесорних систем. Згідно прийнятої термінології мікропроцесорна система — це сукупність апаратних засобів (hardware) і програмного забезпечення (software). Вона розрахована на сумісну роботу з периферійними пристроями: клавіатурою, дисплеєм, принтером, зовнішньою пам'яттю, а також з датчиками і виконавчими механізмами.

Основним елементом системи є *мікропроцесор* — програмно керований цифровий пристрій обробки інформації в мікросхемному виконанні. Завдяки програмному керуванню мікропроцесор реалізує будь-яку залежність між послідовністю вхідних та вихідних сигналів і є універсальним пристроєм. Він працює разом з внутрішньою пам'яттю програм, пам'яттю даних та з пристроями введення/виведення і з'єднаний з ними за допомогою інформаційних каналів – систем шин.

Архітектура всіх мікропроцесорних систем, яка визначає їх структуру і загальну систему логічної організації, має багато спільного. Програма, яку виконує мікропроцесор, знаходиться у комірках пам'яті програм. Мікропроцесор по черзі зчитує кожну команду з пам'яті, аналізує її і потім виконує. Послідовність виконання кожної команди має такий вигляд:

— мікропроцесор виставляє адресу команди на адресну шину;
— вміст комірки пам'яті за вказаною адресою з'являється на шині даних;
— мікропроцесор зчитує команду і виконує її.

Сигнали для реалізації команди передаються по каналам керування. Разом з пам'яттю програм в мікропроцесорних системах передбачається оперативна пам'ять даних для тимчасового розміщення інформації. Модулі введення/виведення забезпечують обмін повідомленнями з периферійними пристроями.

Мікропроцесорні системи діляться за функціональним призначенням на мікро-ЕОМ, які спрямовані на обробку інформації у взаємодії з оператором-користувачем (калькулятори, персональні комп'ютери) та мікроконтролери, що орієнтовані на сумісну роботу з приладами, пристроями та керованими об'єктами в системах автоматичного керування. Така орієнтація визначає певні особливості їх архітектури і конструктивного виконання. Сучасні мікроконтролери випускаються переважно однокристальними і в багатьох випадках є функціонально-завершеними (з вбудованими аналого-цифровими та цифро-аналоговими перетворювачами і широтно-імпульсними модуляторами). Мікроконтролери, які називають також однокристальними мікро-ЕОМ, є головною частиною сучасних автоматичних систем.

1. Класифікація мікропроцесорних систем та їх архітектура

Мікропроцесори (МП) і мікропроцесорні системи (МПС) класифікуються за призначенням, будовою та архітектурою.

За призначенням розрізняють універсальні та спеціалізовані мікропроцесори.Універсальні МП використовують для проведення обчислень, обробки інформації та автоматичного керування.Спеціалізовані МП діляться на сигнальні,мультимедійні та трансп'ютери.

Сигнальні МП призначені для цифрової обробки та фільтрації сигналів в реальному часі, а також обчислення кореляційних функцій та виконання перетворень Фур'є. Мультимедійні МП обробляють аудіосигнали та графічну інформацію, а також використовуються в гральних приставках і побутовій техніці.Трансп'ютери працюють в мультипроцесорних системах і забезпечують паралельну обробку інформації та розподілене керування.

За кількістю великих інтегральних схем (ВІС) розрізняють багатокристальні мікропроцесорні комплекти (МПК) і однокристальні мікроконтролери(ОМК).В МПК можуть використовуватись однокристальні та секційні МП.Однокристальний МП є конструктивно і функціонально завершеним модулем у вигляді однієї ВІС.У секційних МП в одній ВІС реалізується тільки одна окрема функція мікропроцесора, що дозволяє у разі необхідності нарощувати кількість команд.

За типом архітектури розрізняють МП з нейманівською та з гарвардською архітектурою, а за типом системи команд-МП з повним набором команд CISC (Complete Instruction Set Computing) і МП з обмеженим набором команд RISC (Reduced Instruction Set Computing).

Мікропроцесорні системи будуються, виходячи з принципів магістральності, модульності та мікропрограмного керування.

Принцип магістральості полягає в тому, що всі функціональні модулі МПС з'єднуються за допомогою системної шини SB(System Bus), яка утворює однонапрямлену шину адреси AB(Adress Bus), двонапрямленну шину даних DB(Date Bus) та шину (або канали) керування CB(Control Bus).

Принцип модульності полягає в тому ,що мікропроцесорна система будується, виходячи із обмеженої кількості функціонально і конструктивно завершених блоків-модулів, які мають тристабільні входи з високоімпедансним станом CS(Chip Select-вибір кристалу).Завдяки цьому у кожен момент часу до системної шини МПС під'єднані лише два модулі: той, що передає інформацію, і той, що її приймає. Всі інші модулі знаходяться при цьому у високоімпедансному стані.

Принцип мікропрограмного керування надає можливість здійснення елементарних операцій-мікрокоманд (пересилання, додавання, зсуву, логічних оперцій та ін.), певними комбінаціями яких можливо створити потрібні набори команд.

До складу МПС входять такі модулі:мікропроцесор(МП), постійний запам'ятовуючий пристрій (ПЗП), оперативний запам'ятовуючий пристрій(ОЗП), інтерфейс введення/виведення (ІВВ), пристрій введення/виведення (ПВВ),система переривань (СП), а також таймер (Т) і генератор тактових імпульсів (ГТІ), стабілізований кварцовим резонатором QZ (рис.1) .

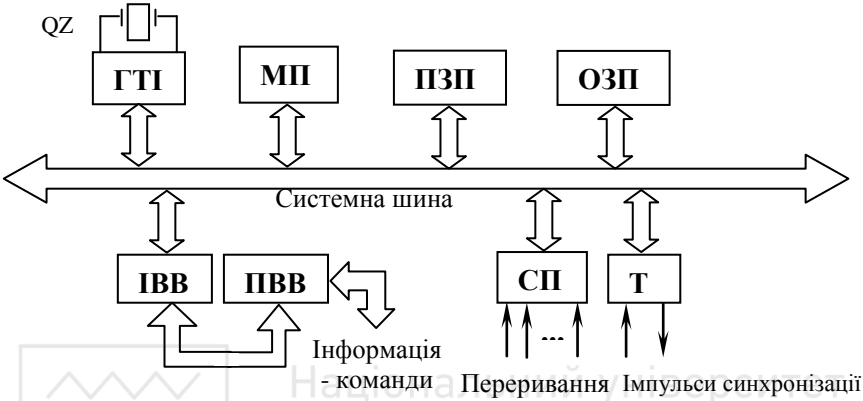


Рис. 1. Узагальнена структурна схема мікропроцесорної системи

Система переривань дозволяє реагувати на зовнішні сигнали-запити переривань, джерелами яких можуть бути сигнали з виходів датчиків. Із появою запиту переривання МП перериває основну програму і переходить до виконання підпрограми обслуговування запиту перереривання. Таймер призначений для реалізації функцій, пов'язаних з відліком часу. МП записує в таймер Т число , яке визначає частоту синхронізації, затримку або коефіцієнт ділення, після чого таймер виконує свої функції самостійно.

Архітектуру мікропроцесора характеризує його структурна схема, програмна модель, ємність пам'яті та способи її адресації, а також опис процедур введення/виведення.

Особливістю неймановської архітектури є розміщення програми і даних у спільній пам'яті MPD (Memory Program/Date), доступ до якої здійснюється по загальній шині команд і даних DB(рис.2)

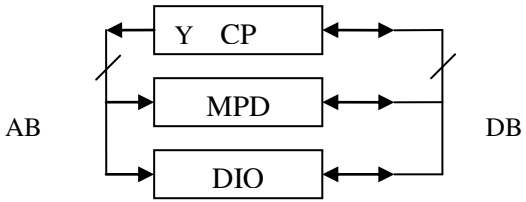


Рис.2. Неймановська архітектура

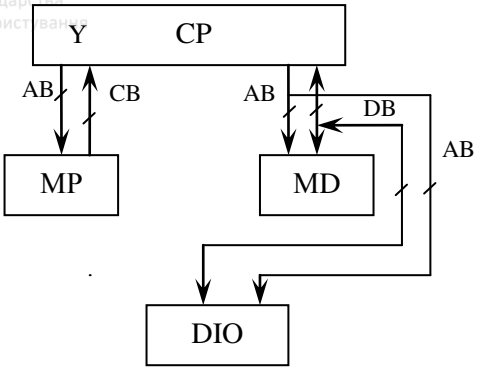


Рис. 3. Гарвардська архітектура

Особливістю гарвардської архітектури є розділення пам'яті програм MP(Memory Program) і пам'яті даних MD(Memory Date) та наявність окремих шин команд (CB) і даних (DB) (рис.3), що забезпечує значне зростання швидкодії.

Структурні схеми обох архітектур містять: мікропроцесорне ядро CP(Central Processor), пам'ять (MPD,MP,MD) та пристрій введення/виведення DIO(Device Input/Output). Всі модулі структурних схем з'єднані за допомогою адресної шини (AB), шини команд-даних (DB) (нейманівська архітектура) або шини команд (CB) та шини даних (DB) (гарвардська архітектура).

Систему пам'яті МПС, що забезпечує запис, зберігання та зчитування інформації, разом з оперативним (ОЗП) та постійним (ПЗП) запам'ятовуючими пристроями (рис.1) створює також регістрова (надоперативна) та інколи внутрішня кеш-пам'ять, що вбудовані безпосередньо в МП. В МП Pentium кеш-пам'ять використовується окремо для команд і даних. Оперативна пам'ять статичного типу будується на тригерах з біполярними або МДН транзисторами (зі структурою метал-діелектрик-напівпровідник), а ОЗП динамічного типу – на ємностях p-n-переходів транзисторів. Вона призначена для запису, тимчасового зберігання і зчитування поточних даних та результатів арифметичних і логічних операцій. Постійна пам'ять МПС працює тільки в режимах зберігання та зчитування кодів команд програм, підпрограм, констант і таблиць.

Разом із вищезгаданою внутрішньою (резидентною) пам'яттю використовують також зовнішню пам'ять на гнучких, твердих і лазерних компакт-дисках (CD-ROM).

2. Характеристики та напрямки розвитку сучасних мікроконтролерів

Сучасні мікроконтролери забезпечують економічні режими роботи, мають досконалу архітектуру, високу швидкодію і надійність. Найпоширенішими серед них є восьмирозрядні контролери сімейства x51.

На сучасному етапі розвитку мікропроцесорної техніки промисловим стандартом вважають однокристальні мікро-ЕОМ фірми *Intel* MCS-51. Всі групи цих мікроконтролерів містять оперативну (128, 256 байт) та постійну (8, 16, 32 Кбайт) пам'ять, двонапрямлені восьмирозрядні порти, вбудовані генератори тактових імпульсів з частотою до 16, 20, 24 МГц, а також інтерфейс для обміну інформацією з іншими мікроконтролерами та персональними комп'ютерами. Їх найбільш досконалі типи мають також програмовані таймери і лічильники, захист внутрішньої пам'яті, широтно-імпульсні модулятори для формування керуючих сигналів та сторожеві таймери WDT (Wathdog Timer), що захищають мікроконтролери від збоїв в роботі. З середини 90-х років фірма *Intel* випускає мікроконтролери MCS-251, які є повністю сумісними з MCS-51, але мають більш досконалу конвейєрну архітектуру та розширену пам'ять.

Фірма *Microchip* випускає восьмирозрядні мікроконтролери PIC16/17 (Peripheral Interface Controller). Вони мають гарвардську архітектуру, електрично програмовану користувачем постійну пам'ять програм (EPROM), оперативну пам'ять даних (RAM), регістри портів введення/виведення, модулі таймерів, а також вбудований аналого-цифровий перетворювач (ADC) та модулі накопичення/порівняння широтно-імпульсного модулятора CCP (Capture/Compare PWM). Висока ефективність сімейств PIC16/17 забезпечується за рахунок використання гарвардської архітектури, при якій пам'ять програм і даних розділені і для звернення до них використовуються окремі шини програм/даних. Це дозволяє збільшити швидкодію і одночасно передавати по шині даних 8-розрядні слова, а по шині команд 14-розрядні коди операцій. Двохступінчатий конвейєр суміщує вибірку чергової команди та виконання поточної. В порівнянні з іншими восьмирозрядними мікроконтролерами аналогічних класів PIC16/17 удвічі зменшують обсяг програм і збільшують швидкодію в чотири рази. Мікроконтролери є універсальними і використовуються у вимірювальних перетворювачах, системах автоматичного контролю, при реалізації будь-яких законів автоматичного регулювання, а також в бортових комп'ютерах автомобілів. Технологія програмованої постійної пам'яті забезпечує зручне і швидке настроювання їх прикладних програм.

Фірма *Atmel* випускає серію мікроконтролерів AT-89, сумісних з промисловим стандартом MCS-51, та мікроконтролери AVR, що мають

гарвардську архітектуру з розділеними шинами даних і програм. Вартість виробів цієї фірми є досить низькою за рахунок електричного програмування пам'яті і відсутності в корпусах мікросхем кварцового скла, яке необхідне при ультрафіолетовому опроміненні комірок пам'яті кристалу. Перевагами мікроконтролерів фірми *Atmel* є наявність вбудованої Flash-пам'яті з достатньо простою процедурою внутрісхемного програмування, вбудованої основної оперативної пам'яті та сторожового таймера. Це забезпечує швидке і високоефективне програмування мікроконтролерів фірми *Atmel* та їх широке застосування у різноманітних галузях промисловості.

Фірма *Zilog* випускає стандартні мікроконтролери Z8 з розширеним набором універсальних функцій; мікроконтролери широкого застосування Z8CCP; мікроконтролери з пониженою напругою живлення Z86Lxx; спеціалізовані мікроконтролери Z86C95 та пристрої для внутрісхемної емуляції.

Мікроконтролери сімейства Z8 мають мультіплексну шину адрес/даних, яка суміщена з портами введення/виведення. Вони містять по три незалежних адресних простори: регістровий файл мікропроцесора (Register File CPU) з адресами регістрів загального призначення, портів введення/виведення та периферії; пам'ять програм (Program memory) з адресами комірок програм і констант та пам'ять даних (Date memory) з адресами комірок даних.Стандартні мікроконтролери Z8 виконують всі основні функції восьмирозрядних мікроконтролерів MCS-51 і PIC 16/17. Мікроконтролери широкого застосування Z8 CCP використовують для створення інтелектуальних датчиків та керування побутовою електронікою.

Мікроконтролери Z86Lxx з пониженою напругою живлення U= 2÷3,6 В споживають 2,4 мВт у робочому режимі та 2 мкВт у режимі очікування і використовуються в об'єктах з обмеженим енергоспоживанням.

Мікроконтролери Z86C95 з цифровим *сигнальним* процесором призначені для цифрової обробки аналогових сигналів у реальному часі і використовуються при автоматичному контролі параметрів об'єктів.

Фірма *Motorola* є одним з провідних виробників мікроконтролерів і пропонує їх велику номенклатуру. Вона є постачальником військово-промислового і аерокосмічного комплексів, а також автомобільної промисловості США. Її мікроконтролери практично вважаються за промисловий стандарт завдяки високій надійності і якості. Разом з тим їх недолік полягає у малій кількості мікроконтролерів з однократно програмованою пам'яттю, що обмежує їх застосування у дрібносерійному виробництві.

Фірма *Philips Semiconductors* випускає більше 100 модифікацій мікроконтролерів сімейства 8051 з тактовими частотами до 40МГц. За останні роки фірма розробила нове мікропроцесорне ядро 8051+, серію високошвидкісних мікроконтролерів, та мікроконтролери з розширеним адресним простором і мікропроцесорним ядром 51MX. Їх основні відміни від

восьмирозрядних мікроконтролерів з класичною архітектурою полягають у збільшенні розрядності програмного лічильника з 16 до 23 розрядів, вказівника стеку з 8 до 16 розрядів та зміни логіки роботи **команд переходу**. Одночасно фірма *Philips* випускає підсімейство мікроконтролерів 51XA з „розширеною архітектурою", яка збільшує їх швидкодію майже у 100 разів порівняно з мікроконтролерами 8051 традиційної архітектури. Це забезпечується головним чином за рахунок збільшення розрядності арифметико-логічного пристрою мікропроцесорного ядра до 16 розрядів, збільшення адресного простору пам'яті програм і даних до 16М і розширеного набору операцій.

Фірма *Dallas Semiconductors-Maxim* випускає серію мікроконтролерів DS500, в яких ОЗП є енергонезалежним завдяки літієвому елементу живлення, що вмонтований безпосередньо у мікросхему і забезпечує десятирічне зберігання інформації в ОЗП мікроконтролерів. Їх мікропроцесорне ядро є ідентичним мікроконтролерам 8051 фірми *Intel*.

Dallas Semiconductors вдосконалила ядро мікроконтролера 8051, внаслідок чого типовий цикл виборки команд зменшився з 12 до 4 тактів і зросла їх швидкодія. Нове сімейство мікроконтролерів має назву High-Speed Microcontroller (високошвидкісні мікроконтролери).

Наступним кроком зростання швидкодії стали мікроконтролери DS89C420 нової групи Ultra High-Speed, в яких час виконання команд у 12 разів менше, ніж в стандартному кристалі сімейства 8051. Важливою особливостю є наявність Flash-пам'яті (16 К) з можливістю внутрісхемного програмування і самопрограмування.

Фірма *Triscend* розробила **конфігуровані** мікроконтролери E5, які складаються з мікропроцесорного ядра базового мікроконтролера 8051, пам'яті і програмованої матриці. Фірма поставляє також пакет розробки, до складу якого входить графічний редактор, компілятор, симулятор та бібліотека периферійних пристроїв. Він забезпечує розробку функціонально-завершених мікроконтролерів заданої конфігурації.

Фірма *Holtec* випускає мікроконтролери, в яких завдяки конвейєру команд одночасно виконується дві операції: виконання поточної команди і вибірка наступної команди з пам'яті. Регістр стану містить флаг нуля (Z), флаг переносу (C), допоміжний флаг переносу (AC), флаг переповнення (OV), флаг економічного режиму (PD) і флаг тайм-аут сторожового таймера (TO). Флаги показують поточний стан мікропроцесорного ядра і впливають на послідовність дій.

Тайванська фірма *Winbond* випускає мікроконтролери із стандартною архітектурою сімейства 8051, які є повністю сумісними з аналогами фірм *Intel*, *Philips*, *Atmel*, але мають підвищену тактову частоту до 40 МГц. ЇЇ останнім досягненням є мікроконтролери Turbo-51, які мають високошвидкісну програмну Flash-пам'ять.

Фірма *Siemens* (*Infineon Technologics*) випускає сімейство мікроконтролерів С500, які є аналогічними мікроконтролерам *Intel* 8051, але мають більш високі тактові частоти. Разом з цим *Siemens* випускає 10-розрядні АЦП, 6-канальний 10-розрядний широтно-імпульсний модулятор для керування електродвигунами постійного струму та інші периферійні пристрої.

Сімейство ICP CON 7000 є розподіленою мікропроцесорною системою і призначене для використання в промислових розподілених системах контролю і обробки даних. До складу сімейства входять керуючий мікроконтролер I-7188 з мікропроцесором AMD і Flash-пам'яттю та сторожевим таймером; локальний контролер реального часу 7188, модуль цифрового введення/виведення з релейним виходом I-7060, аналогові модулі введення/виведення I-7077 та допоміжні перетворювачі RS-232.

З наведеного огляду слідує, що основними напрямками вдосконалення сучасних керуючих однокристальних мікро-ЕОМ-мікроконтролерів є:

- підвищення надійності та ефективності програмування споживачем;
- зменшення енергоспоживання;
- подальше зниження їх собівартості.

3. Системи числення програмного забезпечення мікропроцесорної техніки

Для аналізу процесів, які відбуваються в пристроях мікропроцесорних систем, їх синтезу та програмування необхідно знати правила роботи з числовою інформацією. Діяльність людини пов'язана з використанням десяткової системи числення. При роботі з мікропроцесорною технікою разом з десятковою використовують двійкову та шістнадцяткову системи числення. Будь-яка з них має певний набір цифр і правила їх записування. Загальна кількість усіх цифр системи називається її основою. Поряд написані цифри утворюють числа, причому кожна цифра займає певну позицію.

В десятковій системі числення числа записують десятьма різними цифрами – від 0 до 9, тобто основа системи дорівнює 10. Кожна позиція оцінюється певним значенням: крайня справа нульова позиція (10^0) містить одиниці; наступна перша позиція (10^1) – десятки; друга позиція (10^2) містить сотні і так далі.

Для запису десяткового числа у розгорнутому вигляді кожну цифру в числі множать на значення її позиції (вагу) і всі добутки додають:

$10^3\ 10^2\ 10^1\ 10^0$

$7\ \ 5\ \ 4\ \ 6 = 7\times10^3+5\times10^2+4\times10^1+6\times10^0$;

Якщо десяткове число має дробову частину, то її відокремлюють комою (або крапкою) від цілої частини.

При цьому значення першої справа від коми позиції дорівнює 10^{-1}, наступних позицій – другої і третьої відповідно дорівнює 10^{-2} та 10^{-3} і розгорнута форма запису дробової частини має такий вигляд:

$$\begin{matrix} & 10^{-1} & 10^{-2} & 10^{-3} \\ 0, & 5 & 8 & 9 \end{matrix} = 5\times10^{-1} + 8\times10^{-2} + 9\times10^{-3} = 0{,}5+0{,}08+0{,}009$$

Десяткові числа записують переважно у формі з фіксованою комою, наприклад: 1,35; +9,81; –18,79 і тому подібне. Другим способом їх запису є форма зображення числа з плаваючою комою, при якому різні варіанти запису того ж самого числа визначає місце коми: $365 \rightarrow 3{,}65\times10^{2}$; $36{,}5\times10^{1}$; 3650×10^{-1} і тому подібне. В загальному вигляді число з плаваючою комою записують так: $N=M\times10^{P}$ (М – мантиса; Р - порядок). Мантиса - є нормалізованою, якщо її значення визначається нерівністю виду: $1/q \le M < 1$, де q – основа системи числення.

Використання пакету підпрограм арифметики чисел з плаваючою комою підвищує точність обчислювань при роботі з різноманітними числовими даними.

Мікропроцесорні системи працюють з електричними сигналами, які кодуються двійковими числами, і тому двійкова система числення лежить в основі їх роботи. Двійкова система числення, або система з основою "2" використовує тільки дві цифри "0" та "1". Ці двійкові цифри мають назву бітів (від слів "binari digit" – двійкова одиниця). Фізично біт "0" представляють низькою напругою "L" (LOW), величина якої прямує до "0", а біт "1" – напругою високого рівня певного значення "H" (HIGH), наприклад +5*B*.

Як і у десятковому числі, кожна цифра двійкового числа займає певну позицію. Крайня справа позиція у двійковому числі має значення, що дорівнює 2^0, наступні відповідно 2^1, 2^2, 2^3 і так далі. Десяткове число з двійкового отримують за формулою:

$$N = a_1\times2^{n-1} + a_2\times2^{n-2} + \ldots + a_{n-1}\times2^1 + a_n\times2^0,$$

де a – число, значення якого дорівнює "0" або "1"; n – кількість цифр у двійковому числі (довжина двійкового числа). Наприклад, розгорнуту форму двійкового числа 10110101 можна записати у вигляді: $10110101 = 2^7\times1+2^6\times0+2^5\times1+2^4\times1+2^3\times0+2^2\times1+2^1\times0+2^0\times1$.

Щоб не плутати числа, складені з тих самих цифр, але таких, що належать до різних систем числення, після кожного числа в дужках зазначають систему числення, наприклад $10110101_{(2)} = 181_{(10)}$.

Дробова частина двійкового числа відокремлюється від цілої за допомогою коми, причому кожна позиція справа від коми має таке значення: 2^{-1}, 2^{-2}, 2^{-3} і так далі. Отже, наприклад, розгорнута форма запису двійкового дробового числа:

$$0{,}1101_{(2)} = 1\times2^{-1} + 1\times2^{-2} + 0\times2^{-3} + 1\times2^{-4} = 0{,}5+0{,}25+0+0{,}0625 = 0{,}8125_{(10)}.$$

Двійкові числа, як і десяткові записують у двох формах: з фіксованою і плаваючою комою.

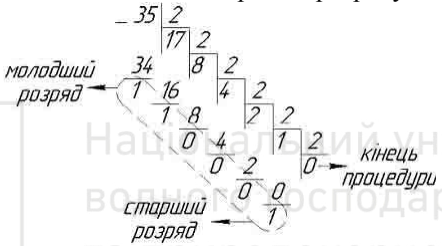
Наприклад: з фіксованою комою

$$N = 111{,}101_{(2)} = 7{,}625_{(10)};$$

з плаваючою комою

$$N = 0{,}111\times10^{101}|_{(2)} = 0{,}875\times2^{5}|_{(10)}.$$

У практичній роботі часто буває потрібно перетворювати числа десяткової системи числення в числа двійкової системи. Для цього ціле десяткове число ділять на основу двійкової системи числення, тобто на 2. Остача від ділення на 2 може дорівнювати 0 або 1. Значення остачі присвоюється молодшому розряду вишукуваного двійкового числа. Результат першого ділення знову ділиться на 2. Остачі ("0" або "1") присвоюється наступний розряд двійкового числа. Подібна процедура повторюється доти, поки результат ділення не буде дорівнювати "0". Остача від останньої дії ділення є значенням старшого розряду двійкового числа:



Отже, $35_{(10)} = 100011_{(2)}$.

При перетворенні десяткового дробу, дробову частину двійкового числа множать на 2. Якщо результат буде менший від 1, то старшому розряду шуканої дробової частини двійкового числа присвоюють значення "0", а якщо більший ніж "1", то "1". Результат попередньої операції знову множать на 2, причому для множення беруть тільки його дробову частину. Аналогічну операцію проводять з новим результатом. Описану процедуру повторюють доти, поки результат наступного множення точно не дорівнюватиме "1", або поки не буде досягнуто необхідної точності.

Розглянемо перетворення десяткового дробу у двійковий на прикладі перетворення числа $0{,}375_{(10)}$:

1. $0{,}375\times2 = 0{,}75 \rightarrow 0$ — старший розряд дробової частини двійкового числа
2. $0{,}75\times2 = 1{,}5 \rightarrow 1$ — наступний розряд дробової частини двійкового числа
3. $0{,}5\times2 = 1 \rightarrow 1$ — молодший розряд дробової частини двійкового числа.

Таким чином, $0{,}375_{(10)} = 0{,}011_{(2)}$.

Якщо десяткове число має дробову частину, то в двійкову систему перетворюють цілу частину згідно першому правилу, а дробову частину – згідно другому і потім записують загальний результат.

Числа у двійковій системі числення займають багато позицій і, щоб спростити їх використання, застосовують шістнадцяткову систему числення (hexadecimal). Її основою є число 16 і вона використовується як засіб скорочення запису двійкових чисел при написанні програм. Шістнадцяткова система містить 16 символів: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E і F, тобто з 0 по 9 – у вигляді цифр, а з 10 по 15 – у вигляді латинських літер, де літера A визначає десять, B – одинадцять і так далі до літери F, яка відповідає п'ятнадцяти. Кожний шістнадцятковий символ може бути визначений, як сполучення 4-х біт. Для перетворення двійкового числа в шістнадцяткове треба поділити число на групи з 4-х біт і замінити їх еквівалентними шістнадцятковими цифрами. Для переводу двійкових чисел у шістнадцяткові та десяткові і навпаки доцільно користуватись таблицею 1:

Таблиця 1

Двійкові числа	Десяткові числа	Шістнадцяткові числа
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Перетворимо двійкове число 1001 1110 в шістнадцяткове. Для цього ділимо двійкове число на групи $1110_{(2)}$ і $1001_{(2)}$. Кожну з них замінюємо відповідним шістнадцятковим числом $E_{(16)}$ і $9_{(16)}$. Тоді шукане шістнадцяткове число дорівнює $1001\ 1110_{(2)} = 9E_{(16)}$.

Навпаки, для перетворення шістнадцяткового числа $7F_{(16)}$ у двійкове число кожну шістнадцяткову цифру замінюють відповідним двійковим числом з 4-х біт:

$$F_{(16)} = 1111_{(2)} \text{ і } 7_{(16)} = 0111_{(2)}.$$

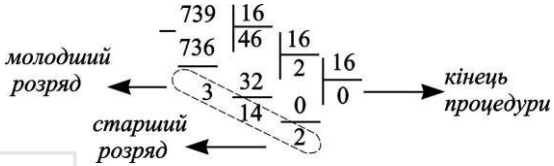
Тоді еквівалентне двійкове число буде дорівнювати $7F_{(16)} = 0111\ 1111_{(2)}$.

Важливим є також перетворення шістнадцяткових чисел у десяткові і навпаки. Розглянемо процес перетворення шістнадцяткового числа 2C6E в десяткове:

$$2C6E_{(16)} = 2\times16^3 + C\times16^2 + 6\times16^1 + E\times16^0 =$$
$$= 2\times4096 + 12\times256 + 6\times16 + 14\times1 = 11374_{(10)}.$$

Значеннями перших чотирьох шістнадцяткових цифр є відповідно 4096, 256, 16 і 1. Десяткове число містить 14 (E_{16}) одиниць, 6 чисел 16^1, 12 (C_{16}) чисел 16^2 і 2 числа 16^3. Сума цих чисел є десятковим числом $11374_{(10)}$.

Процедура перетворення десяткового числа в шістнадцяткове є аналогічною процедурі його перетворення у двійкове число.
Наприклад:



В результаті дістаємо: $739_{(10)} = 2143_{(16)} = 2E3_{(16)}$.

При роботі з мікропроцесорами використовують інколи вісімкову систему числення з основою 8. Існує також спеціальний двійково-десятковий код, в якому кожна десяткова цифра перетворюється у двійковий еквівалент з 4 біт. Наприклад, щоб записати у двійково-десятковому коді число 65, цифри 6 і 5 окремо переводять у натуральний двійковий код: $6 = 110_{(2)}$, $5 = 101_{(2)}$, і потім записують загальний результат: $65_{(10)} = 110101_{(2\text{-}10)}$.

4. Двійкова арифметика

Мікропроцесори виконують арифметичні операції додавання і віднімання, а – також операції множення і ділення двійкових чисел.

Дію двійкового додавання розпочинають з додавання молодших розрядів доданків. Якщо результат додавання більше 1, то до наступного розряду переносять одиницю, а в молодшому розряді пишуть нуль. Потім додають цифри наступних розрядів з урахуванням одиниць, перенесених з попереднього розряду до одержання суми, яку ми шукаємо.

Таблиця додавання двійкових чисел має такий вигляд:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10.$$

З таблиці слідує, що додавання двох одиниць дає нуль у наймолодшому розряді, а одиниця переноситься в наступний старший розряд.

Приклад:

1	1	1	1	1		перенесення
	1	1	1	0	1	перший доданок
+	1	0	1	1	1	другий доданок
1	1	0	1	0	0	сума

$$+\begin{array}{r} 29_{(10)} \\ 23_{(10)} \\ \hline 52_{(10)} \end{array}$$

Дію віднімання розпочинають також з молодших розрядів. Якщо будь-який з розрядів двійкового числа зменшуваного дорівнює 0, а однойменний розряд від'ємника дорівнює 1, то позичають одиницю в сусіднього старшого розряду зменшуваного. Якщо позичено одиницю старшого розряду, то в сусідньому молодшому розряді матимемо дві одиниці.

Таблиця віднімання двійкових чисел має такий вигляд:

$$0 - 0 = 0$$
$$1 - 0 = 1$$
$$1 - 1 = 0$$
$$10 - 1 = 1.$$

Приклад:

		1	10	10				Позика
	1	0	1	0	1	0	1	зменшуване
−		1	1	1	0	0	1	від'ємник
		0	1	1	1	0	$0_{(2)}$	Різниця

$$-\begin{array}{r} 85_{(10)} \\ 57_{(10)} \\ \hline 28 \end{array}$$

(10)

Множення двійкових чисел виконують за правилами, аналогічними для десяткових чисел, тобто визначають проміжні добутки, а потім їх додають.

Таблиця множення двійкових чисел має такий вигляд:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1.$$

Приклад:

```
        x  1  1  0  1           x  1  3
           1  0  0  1                 9
        -------------         ---------
           1  1  0  1           1  1  7 (10)
        0  0  0  0
  +  0  0  0  0
  1  1  0  1
  --------------------
  1  1  1  0  1  0  1 (2)
```

Отже, операція множення двійкових чисел зводиться до операцій зсуву і додавання. Тому в цифрових пристроях мікропроцесорних систем для реалізації операцій множення використовують операції додавання і зсуву.

Ділення – це операція, зворотна множенню. Двійкові числа, як і десяткові, ділять відповідно до правил множення і віднімання двійкових чисел.

Двійкове ділення

```
_11001100 | 1100
 1100     | 10001 (2)
 ----
 _  01
  ___0
 _  011
  ____0
  _  110
   ___0
 _   1100
     1100
     ----
        0
```

Десяткове ділення

```
_204 | 12
  12 | 17 (10)
  --
 _84
  84
  --
  0
```

Реалізувати операцію ділення в мікропроцесорних системах більш складно, ніж операцію множення.

Розглянута вище система запису двійкових чисел за допомогою цифр “0” і “1” є ***натуральним двійковим кодом***.

У мікропроцесорних системах для спрощення математичних операцій віднімання, ділення і деяких інших використовують також інші двійкові коди, які дістають від перетворення натурального двійкового коду. Це прямий, зворотний і додатковий коди.

5. Двійкова арифметика у додатковому коді

Прямі коди використовують при записуванні додатних і від'ємних чисел. Для визначення прямого коду до натурального двійкового коду записують зліва ще один ***знаковий*** розряд.

Якщо число додатне, то знаковим розрядом буде нуль ("0"), а якщо число від'ємне, то знаковим розрядом буде одиниця ("1"). Наприклад, запис додатного числа $39_{(10)}$ прямим двійковим кодом має вигляд:

знаковий розряд	натуральний двійковий код
0	$100111_{(2)}$

,

а запис аналогічного від'ємного числа $-39_{(10)}$ має такий вигляд:

знаковий розряд	натуральний двійковий код
1	$100111_{(2)}$

,

Від'ємні числа доцільно записувати у зворотному і додатковому кодах, що значно спрощує будову функціональних вузлів мікропроцесорних систем. Щоб знайти зворотний код від'ємного десяткового числа, треба спочатку записати його прямим кодом, а потім виконати операцію інверсії, тобто одиниці замінити на нулі і нулі на одиниці в усіх розрядах, крім знакового.

Наприклад, прямий код від'ємного десяткового числа $-123_{(10)}$:

$$1\ \ 1111011_{(2)},$$

а його зворотний код:

$$1\ \ 0000100_{(2)}.$$

Для визначення додаткового коду від'ємного двійкового числа, треба додати до його зворотного коду 1 (інкрементувати число).

Приклад: знайти додатковий код числа $-34_{(10)}$.

Прямий код	1	100010
Зворотний код	1	011101
	+	1
Додатковий код	1	011110

Використовуючи додаткові коди, можливо замінити операцію віднімання на операцію додавання прямого коду зменшуваного з додатковим кодом від'ємника. Така заміна робить можливим універсально використовувати функціональний вузол – суматор, як для додавання, так і віднімання двійкових чисел.

Приклад: $68_{(10)} - 35_{(10)}$.

Прямий код зменшуваного: $68_{(10)} \rightarrow 0\ \ 1000100_{(2)}$,

а додатковий код від'ємника:
прямий код: $-35_{(10)} \rightarrow 1\ \ 0100011_{(2)}$;
зворотний код: $1\ \ 1011100_{(2)}$;
додатковий код: $1\ \ 1011100$

$$+\ \ \underline{\qquad\qquad 1}$$

$$1\ \ 1011101_{(2)}\ .$$

Додамо прямий код зменшуваного до додаткового коду від'ємника. Знакові розряди беруть участь в додаванні як старші розряди. ***Якщо виникає необхідність у переносі одиниці із знакових розрядів, то її відкидають***:

$$\begin{array}{r} 1\ \ 111 \\ 0\ \ 1000100 \\ +\ \ 1\ \ 1011101 \\ \hline (1)0\ \ 0100001_{(2)} \end{array}$$

відкидається

Остаточний результат $0\ \ 0100001_{(2)} = 1\times2^5 + 1\times2^0 = 33_{(10)}$.

знаковий розряд

Найбільш складним є спосіб двійкового ділення, що використовується в мікропроцесорах. При цьому мікропроцесор виконує такі операції:
- додавання дільника в додатковому коді до діленого в прямому коді;
- зсув остачі на кожному кроці ділення на один розряд вліво;
- додавання дільника до попередньо зсунутої остачі.

Приклад: поділити $35_{(10)}$ на $5_{(10)}$.

Натуральні двійкові коди діленого і дільника $35_{(10)} = 100011_{(2)}$ і $5_{(10)} = 101_{(2)}$, а прямі коди діленого і дільника відповідно $35_{(10)} = 0\ \ 100011_{(2)}$ і $-5_{(10)} = 1\ \ 101_{(2)}$. Зворотний код дільника $1\ \ 010_{(2)}$, а його додатковий код $1\ \ 011_{(2)}$.

Процедура ділення методом ***додавання додаткового кода дільника до діленого*** має такий вигляд:

Операції	Біт частки	Найменування операндів
$\begin{array}{r} 0\ \ 100011 \\ +\ \ 1\ \ 011000 \\ \hline 1\ \ 111011 \end{array}$	0	Ділене додатковий код дільника **перша остача**
$\begin{array}{r} 1\ \ 111011 \\ +\ \ 0\ \ 101000 \\ \hline 0\ \ 100011 \end{array}$		перша остача повернене число (дільник) Ділене
$\begin{array}{r} 100011 \\ +\ \ 101100 \\ \hline 0\ \ 01111 \end{array}$	1	ділене після першого зсуву додатковий код дільника **друга остача**

+ 01111 10110 0 0101	1	друга остача після зсуву додатковий код дільника **третя остача**
+ 01010 10110 0 0000	1	третя остача після зсуву додатковий код дільника **четверта остача**

Таким чином: 1 $00011_{(2)} : 101_{(2)} = 0111_{(2)}$

або $35_{(10)} : 5_{(10)} = 7_{(10)}$.

Мікропроцесор починає віднімати дільник з діленого. Якщо він не вкладається у відповідному числі, про що свідчить поява “1” у знаковому біті, то в старшому біті частки записується “0”. Після цього відняті біти дільника повертаються зменшуваному і робиться зсув на один розряд вліво. Потім мікропроцесор знову віднімає дільник і якщо він вкладається у відповідну остачу, про що свідчить поява “0” у знаковому біті, то в наступному біті частки записується “1”. Далі процес продовжується до тих пір, поки остача не стане рівною нулю.

6. Буквенно-цифровий код для обміну інформацією

Для кодування букв, цифр, розділових знаків і керуючих функцій клавіатури в мікропроцесорних системах використовують стандартні буквенно-цифрові коди. Завдяки цьому можливий міжсистемний обмін інформацією без проміжних перетворень. За допомогою буквено – цифрових кодів мікропроцесорні системи взаємодіють також з дисплеєм, принтером і модемом .

Найбільш розповсюдженим є буквенно – цифровий код ASCII (вимовляється як АСКІ) – американський стандартний код обміну інформацією (*American Standard Code for Information Interchange)*. Код ASCII ставить у відповідність кожній великій і рядковій букві, кожній цифрі десяткової системи і розділовому знаку – комбінацію із 7-ми бітів. 8-ий біт використовують для контролю помилок, які можуть виникати при обробці інформації. Сім бітів створюють $2^7 = 128$ двійкових комбінацій. Перші 32 комбінації зарезервовані під спеціальні коди, які використовуються для керування дисплеєм і принтером. Решта 96 кодів мають назву друкованих, тому, що всі вони за винятком першого (“пропуск”) і останнього (“стерти символ”), - відповідають певним символам.

Код ASCII побудований таким чином, що окремі біти у ньому вказують на належність коду до певного класу (наприклад: “великі букви”, “рядкові букви”, “цифри” і т.п.), тоді як інші безпосередньо визначають, яка конкретно це буква або цифра.

Приклад: великій букві "A" відповідає десяткове число $65_{(10)}$, двійковим еквівалентом якого є число $1000001_{(2)}$, а рядкова буква "*a*" в коді ASCII має значення $a_{(ASCII)} = 97_{(10)} = 1100001_{(2)}$. Таким чином різниця між ними міститься в двох крайніх розрядах зліва (10 , 11), які вказують до якого класу належить ця буква, тобто чи є вона великою, або рядковою.

Розглянемо основні коди букв, цифр та розділових знаків ASCII, наведених у таблиці 2. Їх аналіз дає можливість зрозуміти алгоритм побудови ASCII. Слід зазначити, що з незначними змінами цей код використовується в більшості мов.

Таблиця 2

Символ	Код ASCII	Символ	Код ASCII
A	1000001	Z	1011010
a	1100001	z	1111010
B	1000010	0	0110000
b	1100010	1	0110001
C	1000011	2	0110010
c	1100011	3	0110011
D	1000100	4	0110100
d	1100100	5	0110101
E	1000101	6	0110110
e	1100101	7	0110111
F	1000110	8	0111000
f	1100110	9	0111001
…..	……	пропуск	0100000
…..	……	+	0101011
O	1001111	–	0101101
o	1101111	*	0101010
P	1010000	/	0101111
p	1110000	&	0100110
Q	1010001	.	0101110
q	1110001	’	0100111
…..	……	(	0101000
…..	……	)	0101001

7. Логіка цифрових пристроїв

Цифрові мікроелектронні елементи, вузли і пристрої мікропроцесорних систем призначені для роботи з інформацією, яку задано в числовому вигляді. Процес розробки цифрових пристроїв розпочинають з логічного синтезу, результатом якого є складання логічного рівняння, що відображає певну послідовність операцій пристрою. Цю послідовність

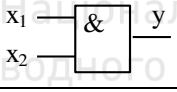
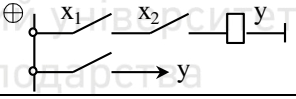
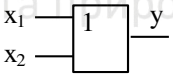
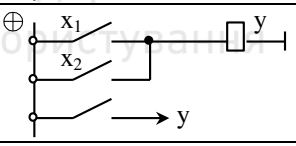
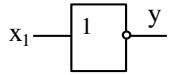
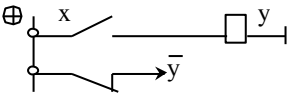
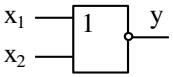
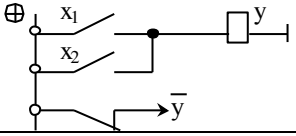
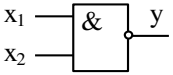
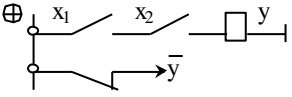
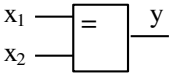
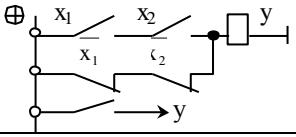
формують на основі алгебри логіки (булевої алгебри), що створена в 1848 р. математиком Джорджем Булем.

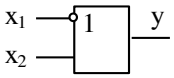
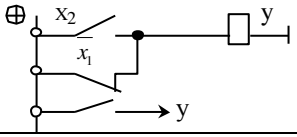
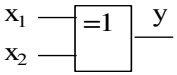
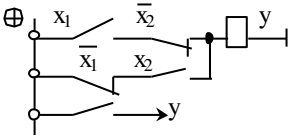
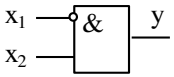
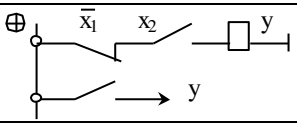
В ній розглядають алгебраїчні залежності, в яких аргументи і функції можуть набувати тільки двох значень “1” і “0”. Практичне використання таких функцій є формальним описом логіки цифрових пристроїв.

Змінна алгебри логіки, яка набуває одне з двох можливих значень і над якою виконують певні дії, є висловлюванням. Висловлювання – це речення, яке може бути істинним або помилковим. За змістом, висловлювання поділяють на прості і складні. В алгебрі логіки над простими висловлюваннями (логічними змінними) виконують дії – логічні операції, які утворюють складні висловлювання.

Використовуючи висловлювання та їхні логічні значення, можливо аналізувати роботу будь – яких релейних схем та цифрових пристроїв. Розглянемо основні логічні функції, їх рівняння, умовні позначення в мікросхемотехніці та релейні еквівалентні схеми, що наведені в таблиці 3:

Таблиця 3

Логічні функції	Умовні позначення	Релейний еквівалент
Кон’юнкція “І” $y = x_1 \cdot x_2$		
Диз’юнкція “АБО”: $y = x_1 + x_2$		
Інверсія “НІ”: $y = \bar{x}$		
Заперечення диз’юнкції (Пірса) “АБО-НІ”: $y = \overline{x_1 + x_2}$		
Заперечення кон’юнкції “І-НІ”: (Шеффера) $y = \overline{x_1 \cdot x_2}$		
Рівнозначність (тотожність): $y = x_1 \cdot x_2 + \overline{x_1 x_2}$		

Імплікація: $y = \overline{x_1} + x_2$		
Виключаюче “АБО”: $y = x_1 \oplus x_2$		
Заборона: $y = \overline{x_1} \cdot x_2$		

Основними логічними операціями є кон’юнкція, диз’юнкція та інверсія.

Кон’юнкцію, або логічну операцію “І” отримуємо при об’єднанні двох і більше простих висловлювань $x_1, x_2, \ldots x_n$ сполучником І. Запис операції кон’юнкція має вигляд $y = x_1 \wedge x_2 = x_1 \cdot x_2$. Операцію І називають також логічним множенням.

Якщо записати всі можливі комбінації значень, яких набувають логічні аргументи x_1 і x_2, і значення логічної функції y для кожної з комбінацій та звести їх у таблицю, то отримаємо таблицю істинності для логічної операції “І” (табл. 4):

Таблиця 4

Аргументи		Функція
x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1

Таблиця істинності 4 є аналогічною таблиці множення двох чисел – “0” і “1”, звідки і назва – логічне множення.

Диз’юнкція, або логічна операція АБО, полягає в об’єднанні простих висловлювань сполучником АБО. Запис операції диз’юнкція

має вигляд $y = x_1 \vee x_2 = x_1 + x_2$. Операцію АБО називають також логічним додаванням. Її таблиця істинності має такий вигляд (табл. 5).

Таблиця 5

Аргументи		Функція
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	1

Інверсія, або логічна операція НІ, є запереченням певного висловлювання x. Якщо висловлювання x є істинним $y = x$, то його заперечення записують як $y = \bar{x}$ (тобто висловлювання $\bar{x}$ буде помилковим). У таблиці істинності ця операція матиме такий вигляд (табл. 6):

Таблиця 6

Аргумент	Функція
x	y
0	1
1	0

Логічні функції І, АБО, НІ є основними в алгебрі логіки тому, що за їх допомогою можливо визначити всі інші логічні функції. Елементи, що реалізують ці функції, називають одноступінчастими. Але базовими елементами мікросхемотехніки є двоступінчасті елементи І - НІ та АБО-НІ.

Функція І-НІ (штрих Шеффера) реалізує логічну операцію заперечення кон'юнкції і її таблиця має такий вигляд (табл. 7):

Таблиця 7

x_1	x_2	y
0	0	1
1	0	1
0	1	1
1	1	0

Функція АБО-НІ (стрілка Пірса) реалізує заперечення диз'юнкції і її таблиця істинності має вигляд (табл. 8) :

Таблиця 8

x_1	x_2	y
0	0	1
1	0	0
0	1	0
1	1	0

Логічна операція рівнозначність означає, що вихідний сигнал існує в тому випадку, якщо на обох входах елемента одночасно є або відсутні логічні одиниці.

Логічна операція імплікація означає, що логічна одиниця на виході логічного елемента існує тільки тоді, коли відсутній сигнал на вході x_1 або є на вході x_2.

Логічна операція виключаюче АБО має таку таблицю істинності (табл. 9):

Таблиця 9

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

Таким чином логічна 1 з'являється на виході елемента тоді, коли вхідні сигнали є різними, тобто один з них має значення логічного 0, а інший-логічної 1.

Логічна операція заборони забезпечує появу логічної 1 на виході тільки при наявності 1 на вході x_2 та 0 на вході заборони x_1 (таблиця 3).

Основні закони алгебри логіки наведені у таблиці 10. Для доведення будь-якого з цих законів треба розглянути всі можливі комбінації значень логічних змінних і виконати над ними відповідні операції.

Таблиця 10

№ п/п	Закони алгебри логіки	
1	Комутативні (переміщувальні)	$x_1 + x_2 = x_2 + x_1;$ $x_1 \cdot x_2 = x_2 \cdot x_1;$
2	Асоціативні (сполучні)	$x_1 + x_2 + x_3 = (x_1 + x_2) + x_3;$ $x_1 \cdot x_2 \cdot x_3 = (x_1 \cdot x_2) \cdot x_3;$
3	Дистрибутивні (розподільні)	$x_1 \cdot (x_2 + x_3) = x_1 \cdot x_2 + x_1 \cdot x_3;$ $x_1 + x_2 \cdot x_3 = (x_1 + x_2) \cdot (x_1 + x_3);$
4	Тавтології (виключення повторення)	$x + x + x = x;$ $x \cdot x \cdot x = x;$
5	Інверсії (де Моргана)	$\overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2};$ $\overline{x_1 \cdot x_2} = \overline{x_1} + \overline{x_2}$
6	Склеювання	$x_1 \cdot x_2 + x_1 \cdot \overline{x_2} = x_1;$
7	Поглинання	$x_1 + \cdot x_1 \cdot x_2 = x_1;$

Із законів випливають такі властивості:

- доповнення: $x \cdot \bar{x} = 0, \quad x + \bar{x} = 1$;
- подвійне заперечення: $\bar{\bar{x}} = x$;
- додавання з одиницею: $x + 1 = 1$;
- додавання з нулем: $x + 0 = x$;
- множення на одиницю: $x \cdot 1 = x$;
- множення на нуль: $x \cdot 0 = 0$;

Логічну функцію можна задати трьома способами: словесним, табличним і аналітичним.

Таблиця істинності – це табличний спосіб задання логічної функції. В ній записують усі можливі комбінації значень логічних аргументів і значення функції для кожної з комбінацій.

При аналітичному способі логічну функцію записують рівнянням, що виводиться з таблиці істинності. Схеми мікропроцесорних систем та їх окремих елементів розробляють на основі заздалегідь складених логічних рівнянь.

Існують дві форми запису логічної функції в аналітичному вигляді:

– досконала диз'юнктивна нормальна форма (ДДНФ);
– досконала кон'юнктивна нормальна форма (ДКНФ).

В цих формах кожна елементарна кон'юнкція або диз'юнкція містить всі змінні (з інверсіями або без них) і відсутні однакові кон'юнкції або диз'юнкції.

При складанні логічних рівнянь в ДДНФ треба зробити наступне:

- з таблиці істинності вибрати рядки аргументів x_i, яким відповідають логічні функції, що дорівнюють одиниці ($y_i = 1$);
- записати логічні добутки аргументів кожного з обраних рядків і з'єднати їх знаком логічної суми;
- над аргументами, що дорівнюють нулю, поставити знак інверсії ($\bar{x}_i = 0$).

Розглянемо як приклад виведення в ДДНФ логічного рівняння операції виключаючого АБО. Для цього вибираємо з її таблиці істинності (табл.. 9) набори аргументів 2-го і 3-го рядків, в яких логічна функція дорівнює 1, і далі виконуючи правила складання логічного рівняння в ДДНФ, отримуємо:

$$x_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2 = y \cdot$$

При складанні логічних рівнянь в ДКНФ необхідно виконати наступні операції:

- з таблиці істинності вибрати рядки аргументів x_i, яким відповідають логічні функції, що дорівнюють нулю ($y_i = 0$);
- записати логічні суми аргументів кожного обраного рядка і з'єднати їх знаком логічного добутку;
- над аргументами, що дорівнюють одиниці, поставити знак інверсії ($\bar{x}_i = 1$).

Якщо записати логічну функцію виключаючого АБО в ДКНФ, то її рівняння матиме наступний вигляд:

$$(x_1 + x_2) \cdot (\bar{x}_1 + \bar{x}_2) = y \cdot$$

Слід зазначити, що логічні рівняння для тієї самої функції, що записані у ДДНФ і ДКНФ мають різний вигляд, але є рівнозначними:

$$y = (x_1 + x_2) \cdot (\bar{x}_1 + \bar{x}_2) = x_1 \cdot \bar{x}_1 + x_1 \cdot \bar{x}_2 + x_2 \cdot \bar{x}_1 + x_2 \cdot \bar{x}_2 = x_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2.$$

Використовувати ДДНФ доцільно, якщо кількість наборів з $y = 0$ більша ніж з $y = 1$.

Визначення логічної функції, яка описує принцип дії схеми цифрового пристрою, здійснюється за допомогою логічного синтезу.

Логічний синтез складається з таких операцій:

- виходячи з опису роботи цифрового пристрою визначають його логічні аргументи та функцію і складають відповідну таблицю істинності на першому етапі;
- виводять логічне рівняння в ДДНФ або ДКНФ;
- виконують мінімізацію отриманого логічного рівняння.

Розглянемо операції логічного синтезу на такому прикладі:

вивести логічне рівняння цифрового пристрою, який має три входи x_1, x_2, x_3 і сигнал на виході якого „y" з'являється при наявності хоча б двох вхідних сигналів.

Для цього складемо таблицю істинності (табл. 11), що відповідає такому словесному опису:

Таблиця 11

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Потім, виконуючи правила виведення логічного рівняння в ДДНФ, отримуємо:

$$\overline{x}_1 \cdot x_2 \cdot x_3 + x_1 \cdot \overline{x}_2 \cdot x_3 + x_1 \cdot x_2 \cdot \overline{x}_3 + x_1 \cdot x_2 \cdot x_3 = y \cdot$$

Операція мінімізації отриманого рівняння розглянута у наступному розділі.

8. Мінімізація логічних функцій

Мінімізація – це процес визначення мінімальних форм логічних рівнянь, складених в досконалій нормальній формі (ДДНФ або ДКНФ). Їх спрощення дозволяє зменшити кількість елементів цифрових мікросхем і забезпечити синтез оптимальних мікропроцесорних систем.

Якщо кількість аргументів – вхідних змінних логічних рівнянь $n \leq 6$, то для мінімізації використовують карти Карно. Їх будують у вигляді таблиць з кількістю клітинок 2^n , причому у кожній з них знаходиться одна комбінація всіх вхідних змінних (мінтерм). Всі сусідні клітинки відрізняються значенням тільки однієї змінної. Таким чином кожному набору вхідних змінних в карті Карно відповідає клітинка, а не рядок, як в таблиці істинності, і тому вона є більш компактним зображенням логічної функції. Вхідні величини діляться на дві групи і їх значення приписуються відповідно рядкам карти та її стовпцям.

Розглянемо карту Карно для функції трьох вхідних змінних (n=3): x_1, x_2, x_3 (табл.. 12). При цьому її рядки відмічені значеннями аргумента x_1, а стовпці значеннями аргументів x_2, x_3.

Таблиця 12.

x_2x_3 / x_1	00	01	11	10
0	$\overline{x_1}\cdot\overline{x_2}\cdot\overline{x_3}$	$\overline{x_1}\cdot\overline{x_2}\cdot x_3$	$\overline{x_1}\cdot x_2\cdot x_3$	$\overline{x_1}\cdot x_2\cdot\overline{x_3}$
1	$x_1\cdot\overline{x_2}\cdot\overline{x_3}$	$x_1\cdot\overline{x_2}\cdot x_3$	$x_1\cdot x_2\cdot x_3$	$x_1\cdot x_2\cdot\overline{x_3}$

Кількість клітинок таблиці відповідає кількості комбінацій вхідних величин і дорівнює $2^3 = 8$. Кожна її клітинка заповнена мінтермами функції $y=f(x_1, x_2, x_3)$.

Мінімізуємо, як приклад функцію, отриману з таблиці істинності 11 і записану в досконалій диз'юнктивній нормальній формі (ДДНФ). Для цього в таблиці 12 позначимо „1" клітинки, що відповідають членам рівняння $\overline{x_1}\cdot x_2\cdot x_3$, $x_1\cdot\overline{x_2}\cdot x_3$, $x_1\cdot x_2\cdot\overline{x_3}$, $x_1\cdot x_2\cdot x_3$, а в решту клітинок запишемо „0". Тоді карта Карно матиме вигляд (табл.. 13).

Таблиця 13.

x_2x_3 / x_1	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Склеюючи мінтерми $\overline{x_1}\cdot x_2\cdot x_3$ і $x_1\cdot x_2\cdot x_3$ по змінній x_1 і об'єднуючи їх в один контур, отримуємо кон'юнкцію двох змінних: $x_2\cdot x_3(\overline{x_1}+x_1)=x_2\cdot x_3\cdot 1=x_2\cdot x_3$. Аналогічно для мінтермів $x_1\cdot\overline{x_2}\cdot x_3$ і $x_1\cdot x_2\cdot x_3$ склеювання відбувається по змінній x_2: $x_1\cdot x_3(\overline{x_2}+x_2)=x_1\cdot x_3\cdot 1=x_1\cdot x_3$, а для мінтермів $x_1\cdot x_2\cdot x_3$ і $x_1\cdot x_2\cdot\overline{x_3}$ - по змінній x_3: $x_1\cdot x_2(x_3+\overline{x_3})=x_1\cdot x_2\cdot 1=x_1\cdot x_2$.

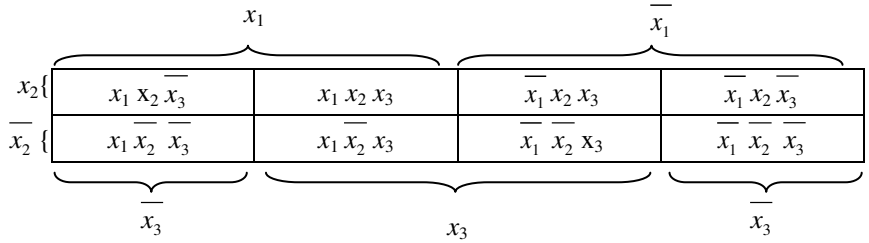
В результаті отримаємо мінімізоване рівняння логічної функції:

$$Y=x_1\cdot x_2+x_2\cdot x_3+x_1\cdot x_3$$

Для мінімізації логічних функцій використовують також діаграми Вейча, які є аналогічними картам Карно, але відрізняються від них способом розмітки: замість символів „0" і „1" використовують позначення аргументів $x_1,\overline{x_1}$; $x_2,\overline{x_2}$; $x_3,\overline{x_3}$ і т.п.

Розглянемо метод мінімізації Вейча також на прикладі трьох аргументів. При цьому кількість всіх комбінацій становитиме 8. Діаграма Вейча для трьох аргументів має такий вигляд (табл. 14):

Таблиця 14



Щоб спростити процес знаходження потрібної комірки, на краях таблиці записують значення змінних згідно наведеній схемі і у кожну комірку таблиці заносять значення однієї комбінації

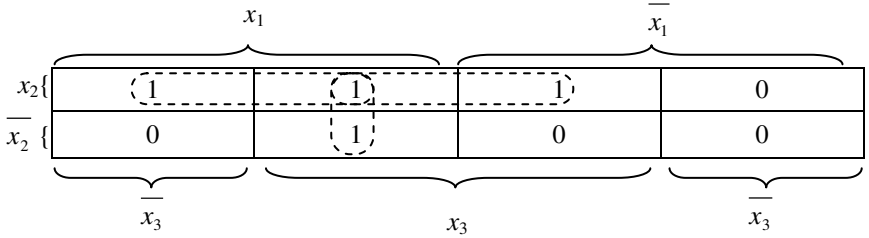
Наступний крок – це завдання в таблиці логічного рівняння в ДДНФ, яке треба мінімізувати. Для цього одиницями позначають комірки, які відповідають членам, що є в цьому рівнянні. Решту комірок позначають нулями.

Наприклад, функція трьох змінних, що отримана з таблиці істинності 11:

$$y = x_1\, x_2\, \overline{x_3} + x_1\, \overline{x_2}\, x_3 + \overline{x_1}\, x_2\, x_3 + x_1\, x_2\, x_3,$$

на діаграмі Вейча матиме такий вигляд:

Таблиця 15



Якщо функція містить “1” в сусідніх комірках, то їх об’єднують в контури. В один контур можна об’єднати 2 або 4 сусідні “1”. Під сусідніми в таблиці для трьох змінних , як і в картах Карно, розуміють також “1”, що стоять на протилежних бічних краях таблиці.

За такого об’єднання члени рівняння, що позначені в таблиці сусідніми одиницями, в кінцевому мінімізованому рівнянні матимуть вигляд логічного добутку, в якому немає змінної, що змінює знак для даного об’єднання.

Сума таких добутків, що відповідає мінімальному числу контурів, якими можна охопити всі сусідні одиниці, і буде кінцевим мінімізованим рівнянням.

У наведеному прикладі в першому контурі змінна x_3 змінюється на $\overline{x_3}$, отже цьому контуру відповідає логічний добуток $x_1 \cdot x_2$. У другому контурі змінна x_1 змінюється на $\overline{x_1}$ і цьому контуру відповідає добуток $x_2 \cdot x_3$. Третій контур відповідає добутку $x_1 \cdot x_3$, бо змінна x_2 змінюється на $\overline{x_2}$. Таким чином мінімізоване рівняння має вигляд:

$$y = x_1x_2 + x_2x_3 + x_1x_3$$

і результат мінімізації співпадає з попереднім результатом, отриманим за допомогою карт Карно.

Якщо функція, яку задано в таблиці, не має сусідніх одиниць, то вона не мінімізується і далі треба застосовувати вихідне логічне рівняння.

9. Синтез цифрових пристроїв

9.1. Загальна структура цифрових пристроїв, скінчені автомати та способи їх опису

Комбінаційні цифрові пристрої будують на логічних елементах, а послідовні цифрові пристрої містять також елементи пам'яті (тригери). Для їх аналізу і синтезу застосовується **теорія цифрових автоматів**.

Цифровий автомат це пристрій, призначений для обробки дискретної інформації, причому вхідні та вихідні величини X , Z визначені у дискретні проміжки часу. Цифровими автоматами є тригери, функціональні вузли і модулі мікропроцесорних систем (великі інтегральні схеми), а також мікропроцесорні системи. Будь-який реальний цифровий пристрій може знаходитись лише у певних дискретних станах (наприклад "0" або "1") і має фіксований обсяг пам'яті. Тому математичною моделлю цифрового автомата є скінчений автомат (finite state). При вивченні скінчених автоматів використовують їх абстрактну і структурну теорію.

Абстрактні автомати – це скінчені автомати, елементи яких X, Z, A є символами вхідних, вихідних та внутрішніх змінних. При цьому визначають загальні закони їх функціонування без врахування структур. Величини, що характеризують автомат, представляють алфавітним способом: $X = (x_1, x_2, x_3, \ldots x_n)$ – вхідний алфавіт; $Z = (z_1, z_2, z_3, \ldots z_n)$ – вихідний алфавіт; $A = (a_1, a_2, a_3, \ldots a_n)$ – алфавіт внутрішніх станів.

Скінчений автомат, вихідні сигнали якого є функціями його внутрішніх станів A та вхідних сигналів X, називається автоматом Мілі:

$$A_{(n)} = f\,[A_{(n-1)}, X_{(n)}];$$
$$Z_{(n)} = \varphi\,[A_{(n-1)}, X_{(n)}],$$

де n – дискретний момент часу.

Автомат Мура це скінчений автомат, робота якого описується такими рівняннями:

$$A_{(n)} = f\,[A_{(n-1)}, X_{(n)}];$$
$$Z_{(n)} = \varphi\,[A_{(n)}],$$

тобто, вихідні сигнали цього автомата залежать тільки від його внутрішніх станів.

Основними способами опису скінчених автоматів є аналітичний у вигляді рівнянь, табличний у вигляді таблиць істинності і переходів та графи автоматів. Таблиця переходів (станів) – це матриця, кожному внутрішньому стану якої відводиться один рядок, а кожній вхідній величині – один стовпчик. Елемент a_j матриці є внутрішнім станом, в який переходить автомат зі стану a_i (і навпаки) при певних вхідних сигналах.

Графами автоматів є діаграми переходу, в яких множинам вершин відповідають вершини внутрішніх станів, а множини ребер (дуг) – множинам можливих переходів з одного стану в інші.

Як згадувалось вище, скінчені автомати діляться на комбінаційні та послідовні. Комбінаційні автомати описуються рівняннями $Z = f(X)$, тобто системою логічних функцій і мають незмінний внутрішній стан. Їх вихідні стани повністю визначаються сукупністю вхідних станів у даний момент X_n і не залежать від станів сигналів, що надійшли у попередній момент часу. Аналіз і синтез цих автоматів розглянутий в розділах 7,8, присвячених логічному синтезу та мінімізації логічних функцій. До комбінаційних автоматів відносяться функціональні вузли: шифратори, дешифратори, напівсуматори, суматори, мультиплексори, демультиплексори, компаратори та комбінаційні зсувачі. Послідовні автомати описуються рівняннями, які враховують як вхідні величини X, так і внутрішні стани A. До них належать регістри, лічильникі, запам'ятовуючі пристрої, а також модульні пристрої мікропроцесорних систем та мікропроцесорні системи.

В кожний момент часу t послідовний скінчений автомат знаходиться у певному стані $a^n = a(t)$ із множини станів A. Послідовність вхідних сигналів $x^n = x(t)$ із алфавіту X, що надходять у дискретні проміжки часу, створює вхідне слово. Скінчений автомат ставить у відповідність кожному з них послідовність вихідних сигналів $z^n = z(t)$ із алфавіту Z. Таким чином автомат реалізує певне відображення, яке однозначно визначається заданим значенням функції переходів a_i та функції входів x_i.

Наприклад, якщо автомат функціонує за таблицею істинності 16а, то він є автоматом Мілі, заданим у вигляді таблиць переходів 16б, причому значення вихідної величини z^{n-1} характеризує стан автомата в момент часу t_1, а z^n – в момент часу t_2.

Таблиця 16а. Таблиця істинності

Входи		Вихід
x_1	x_2	z^n
0	0	z^{n-1}
0	1	1
1	0	0
1	1	-

Таблиця 16б. Таблиця переходів

Входи		Стани	
x_1	x_2	a_0	a_1
0	0	a_0	a_1
0	1	a_1	a_1
1	0	a_0	a_0
1	1	-	-

На рис.4 наведений граф автомата, причому його вузлам відповідають стани автомата a_0 або a_1, а дуги між вузлами визначають переходи із одного стану в інший під впливом вхідних сигналів x_1, x_2.

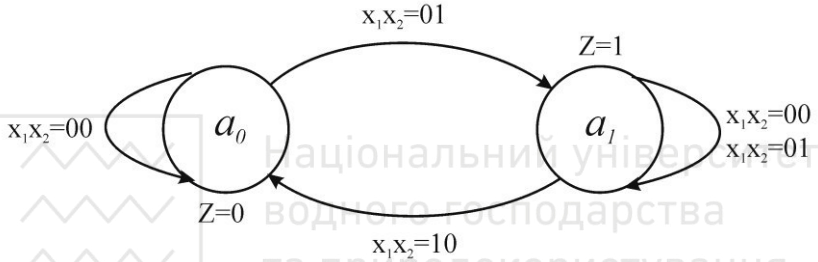


Рис. 4. Граф автомата з двома станами

При початковому стані a_0 і наборі вхідних величин $x_1x_2 = 00$ стан автомата a_0 зберігається, і його вихідний сигнал $z = 0$. Якщо набір вхідних сигналів $x_1x_2 = 01$, то стан автомата змінюється з a_0 на a_1, а вихідний сигнал із $z = 0$ на $z = 1$. Аналогічно аналізується функціонування автомата для початкового стану a_1.

На структурному етапі автомат представляють у вигляді структурної схеми із елементів стандартного набору. Основою для побудови автомата є алгоритм його функціонування. Розглянемо методику побудови графа алгоритму автомата з трьома станами. Для цього вводиться множина $У=\{y_0, y_1,...y_p\}$, яка відображає мікрокоманди, що спричиняють переходи автомата з одного стану в інший.

Щоб отримати з графа алгоритму роботи (рис. 5) граф автомата Мілі треба позначити його вершини як стани автомата a_1, a_2, a_3, а дуги - у вигляді вхідних функцій x_1, x_2, x_3 та функцій переходу y_1, y_2, y_3, y_4 (рис. 6).

Для побудови графа автомата Мура, еквівалентного автомату Мілі (рис. 6), символами a_i (вершини графа) позначають вершини мікрокоманд y_i. При цьому приймають, що $y_0 = y_p = a_1$, а для інших операційних вершин приймають, що $y_1 = a_2$; $y_2 = a_3$; $y_3 = a_4$; $y_4 = a_5$; $y_5 = a_6$; (рис. 7). Вхідні сигнали

x_1, x_2, x_3, що забезпечують зміни сигналів автомату, використовуються як функції переходу між операційними вершинами (мікрокомандами).

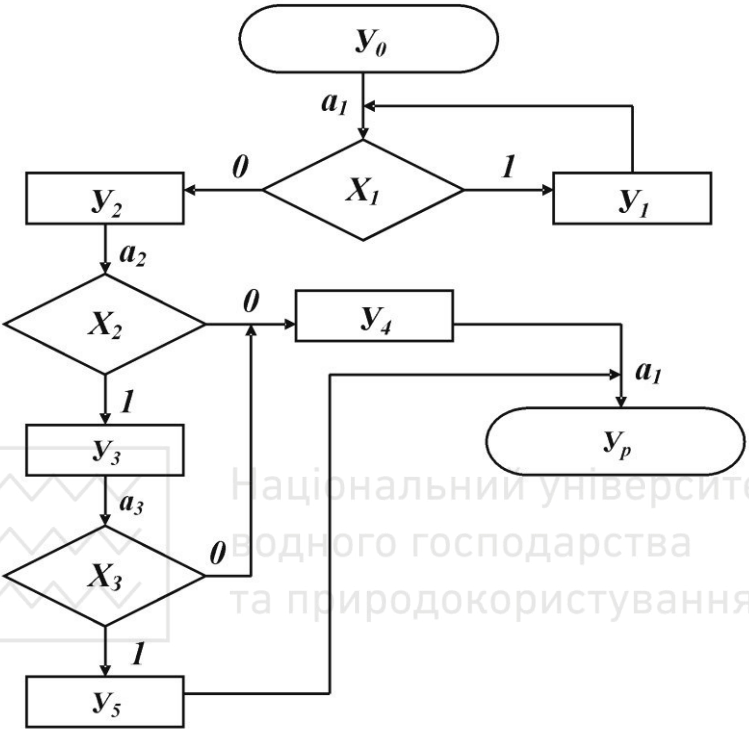


Рис. 5. Граф алгоритму автомата з трьома станами

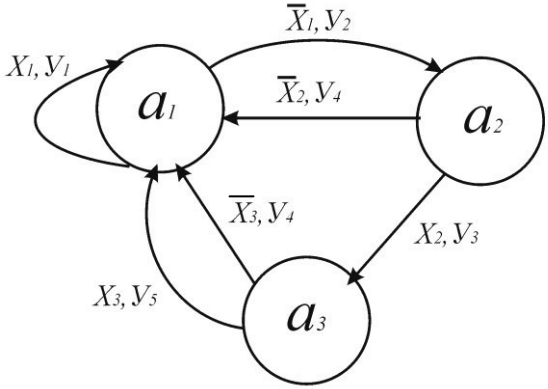


Рис. 6. Граф автомата Мілі, отриманий з графа алгоритму його роботи

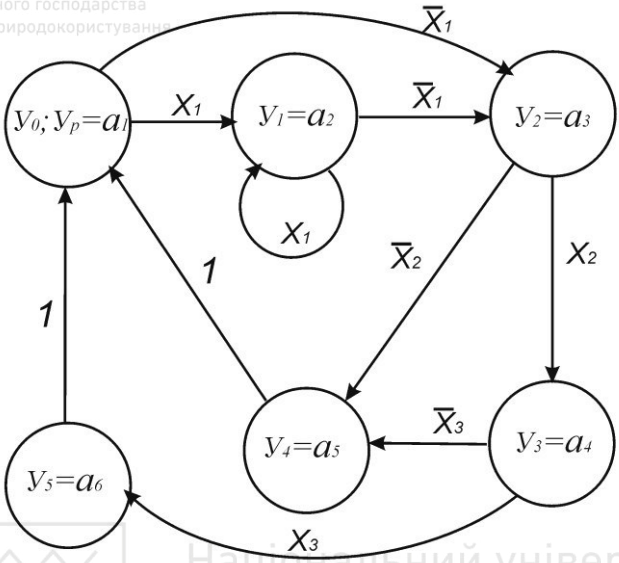


Рис. 7. Граф автомата Мура, отриманий з графа алгоритму його роботи

9.2. Складання структурних схем послідовних цифрових пристроїв на елементах пам'яті та логічних елементах

Після формального опису автомата його реалізовують з використанням елементів мікросхемотехніки. Розглянемо як приклад схему цифрового автомата зі зворотним зв'язком (рис.8):

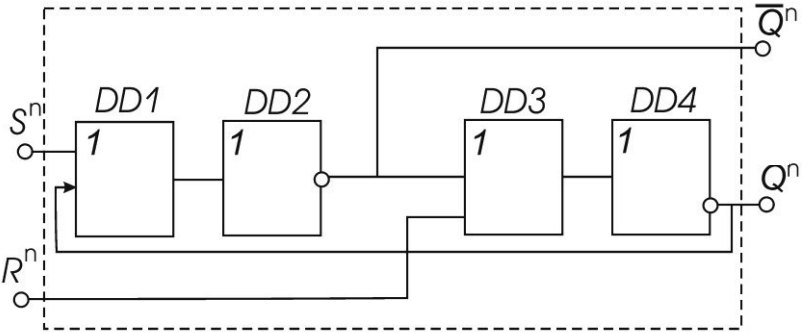


Рис. 8. Структурна схема цифрового автомата зі зворотним зв'язком

Його вхідними сигналами в n – му такті є S^n; R^n, вихідними сигналами - $Q^n; \overline{Q^n}$, а вихідними сигналами в попередньому n-1 – му такті є $Q^{n-1}; \overline{Q^{n-1}}$. Функціонування автомата відображено у таблиці переходів, де наведено всі можливі стани автомата при різних комбінаціях вхідних сигналів, та сигналу, що надходить на вхід пристрою через коло зворотного зв'язку.

Таблиця 17. Таблиця переходів цифрового автомата зі зворотним зв'язком

№	Значення вхідних станів			Вихідний сигнал	Опис стану
п/п	S^n	R^n	Q^{n-1}	Q^n	автомата
1	0	0	0	0	Зберігання "0"
2	0	0	1	1	Зберігання "1"
3	0	1	0	0	Підтвердження "0"
4	0	1	1	0	Встановлення "0"
5	1	0	0	1	Встановлення "1"
6	1	0	1	1	Підтвердження "1"
7	1	1	0	F	Невизначеність
8	1	1	1	F	Невизначеність

Аналіз його роботи доводить, що при $Q^{n-1} = 0$ ($\overline{Q}^{n-1} = 1$) і надходженні вхідних сигналів $S^n = 0$; $R^n = 1$, - попередні значення вихідних сигналів зберігаються: $Q^n = 0$ ($\overline{Q}^n = 1$), а при вхідних сигналах $S^n = 1$; $R^n = 0$, - формуються нові вихідні сигнали: $Q^n = 1$ ($\overline{Q}^n = 0$). Якщо $Q^{n-1} = 1$ ($\overline{Q}^{n-1} = 0$), то при надходженні вхідних сигналів $S^n = 0$; $R^n = 1$ на прямому і інверсному виходах схеми формуються нові значення сигналів: $Q^n = 0$ ($\overline{Q}^n = 1$), а при $S^n = 1$; $R^n = 0$, - зберігаються попередні значення вихідних сигналів: $Q^n = 1$ ($\overline{Q}^n = 0$). Але, якщо на обидва входи одночасно надходять сигнали $S^n = R^n = 1$, то функціонування автомата порушується і на обох виходах виникає факультативний стан невизначеності ("F"). Рівняння автомата має такий вигляд: $Q^n = S^n + \overline{R^n} \cdot Q^{n-1}$. Розглянутий автомат повністю співпадає з автоматом Мура.

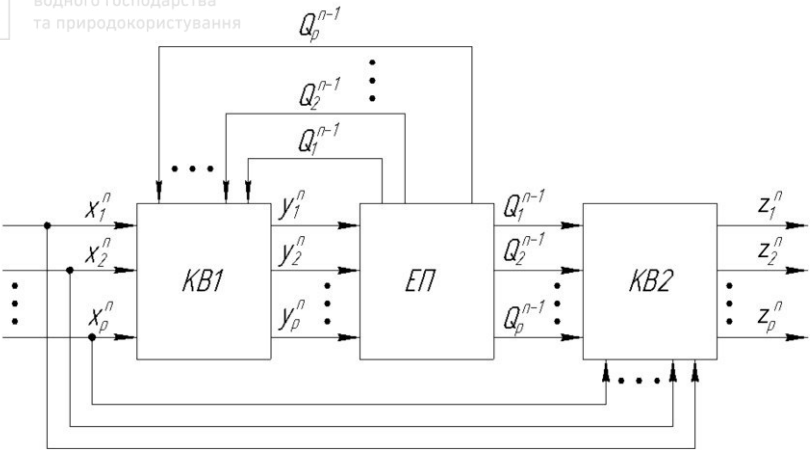


Рис. 9.Узагальнена структурна схема цифрового автомата: ЕП – елементи пам'яті – тригери; КВ1,КВ2 – комбінаційні вузли.

Як згадувалося вище, цифрові автомати разом з елементами пам'яті містять також комбінаційні функціональні вузли на логічних елементах. Їх схеми синтезують згідно з даними таблиць переходів. Узагальнена структурна схема цифрового автомата наведена на рис.9. Щоб отримати p внутрішніх станів автомата, необхідно передбачити у його складі $k \geq \log_2^p$ тригерів ($p \leq 2^k$). Значення керуючих сигналів елементів пам'яті y_i^n (i = 1,2,...p) в n–му такті визначаються вхідними сигналами цифрового автомата x_i(i = 1,2,...p) в n–му такті та станами елементів пам'яті Q_j^{n-1} (j = 1,2,...*p*) в n-1–му такті. Вихідні величини z_j^n (j = 1,2,...*m*) в n–му такті визначаються вхідними сигналами x_i^n (i = 1,2,...p) та станами елементів пам'яті Q_j^{n-1} (j = 1,2,...*p*). Кількість входів та виходів цифрового автомата визначається відповідно як $k_{ВХ} = \log_2^n$ та $k_{ВИХ} = \log_2^m$, причому n,m – кількість вхідних та вихідних кодових комбінацій.

Як наведено вище, прикладом скінченого цифрового автомата є мікропроцесор, який здійснює прийом, зберігання та перетворення дискретної інформації згідно певному алгоритму. *Він синтезується у вигляді двох пристроїв: операційного та керуючого*. Операційний пристрій (ОП) виконує арифметичні, логічні та допоміжні операції. Він містить регістри, шифратори, дешифратори, суматори, лічильники, канали передачі інформації та мультиплексори для комутації каналів. Керуючий пристрій (КП)

координує функціонування вузлів ОП і виробляє у певній часовій послідовності керуючі сигнали, під дією яких вузли ОП виконують необхідні дії.

Процес функціонування ОП складається із послідовності таких основних *елементарних дій в його вузлах*:

- встановлення регістрів в певний стан (наприклад, записування в регістр $R1$ числа 0: $R1 \leftarrow 0$);
- пересилання вмісту одного вузла в інший (наприклад, пересилання вмісту регістра $R2$ в регістр $R1$: $R1 \leftarrow (R2)$);
- зсув вмісту вузла вліво або вправо (наприклад, зсув на 1 розряд вліво регістру R1: $R1 \leftarrow RLC(R1)$);
- рахунок, при якому число в регістрі лічильника зростає або зменшується на 1 ($n \leftarrow (n \pm 1)$);
- порівняння: $R1 \leftarrow CMP(R2)$;
- додавання $R1 \leftarrow [(R2) + (R1)]$;
- основні логічні операції: кон'юнкція (І), диз'юнкція (АБО), інверсія (НІ), які виконуються порозрядно.

Кожна така *елементарні дія*, що виконується *в одному з вузлів на протязі одного тактового періода, є мікрооперацією. Сукупність одночасно виконуваних мікрооперацій називається мікрокомандою*, а набір мікрокоманд, спрямований на розвя'зок певної задачі, – *мікропрограмою*.

КП виконує функції мікропрограмного автомата, тобто визначає, яка поточна мікропрограма повинна надходити на вузли ОП, і забезпечує її передачу по відповідним колам.

10. Принципи побудови мікропроцесорів

Існують два принципи побудови керуючих пристроїв мікропроцесорів: це *принципи схемної та програмованої логіки*.

Синтез мікропроцесорів з використанням *схемної логіки* грунтується на синтезі модульних керуючих пристроїв $КП_М$, які реалізують окремі типові мікрооперації (наприклад, множення, ділення і т.п.). Операція, що реалізується в мікропроцесорі, представляється відповідною командою, код якої за допомогою дешифратора перетворюється на сигнали, які вмикають відповідні модульні пристрої мікрооперацій (рис.10). При цьому алгоритм розв'язку задачі має вигляд послідовності команд, що відповідають типовим мікрооперацiям. Ця послідовність команд створює програму, яка зберігається в пам'яті. Зчитування команд з пам'яті та їх виконання в операційному пристрої забезпечує розв'язок поставлених задач.

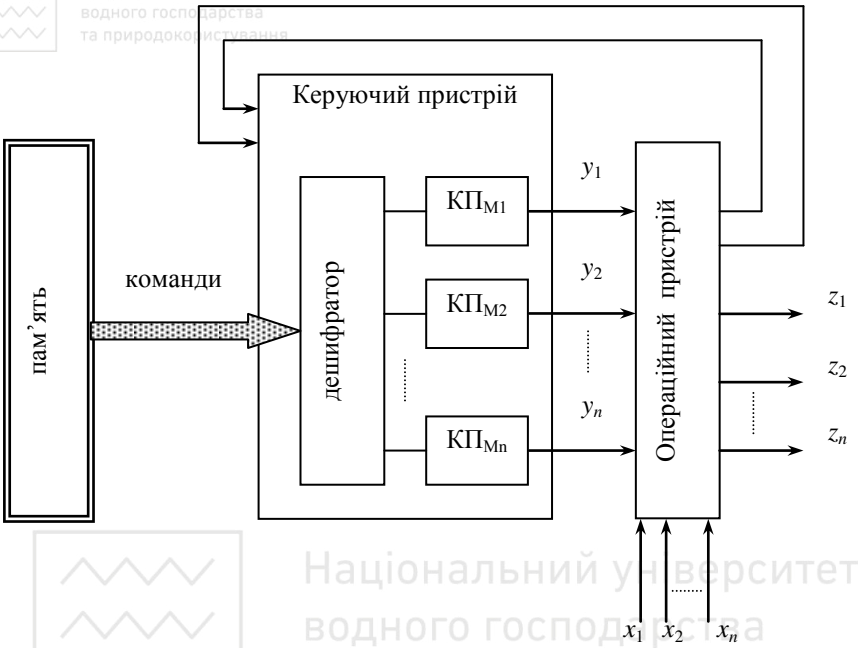


Рис. 10. Структурна схема мікропроцесора
зі схемною логікою

Синтез мікропроцесора з використанням принципа схемної логіки починають з синтезу операційного пристрою (ОП). Виходячи з алгоритму відповідної мікрооперації, складається структурна схема ОП. Після цього проводиться синтез керуючого пристрою (КП) в формі цифрових автоматів Мілі або Мура.

Для реалізації принципу *програмованої логіки на виході керуючого пристрою формуються кодові комбінації мікрокоманд.* Коди мікрокоманд зберігаються у спеціально призначеній для цього *керуючій пам'яті.* При виконанні операції мікрокоманди мікропрограми послідовно зчитуються з керуючої пам'яті і формують сигнали керування операційним пристроєм. Такий спосіб реалізації операцій називається мікропрограмним, а керуючий пристрій -керуючим пристроєм з програмованою логікою (рис.11). Керуючий пристрій разом з керуючою пам'яттю містить також блок мікропрограмного керування, який визначає адресу чергової мікрокоманди в керуючій пам'яті. Адресу першої мікрокоманди мікропрограми задає код операції, що надходить з регістра команд.

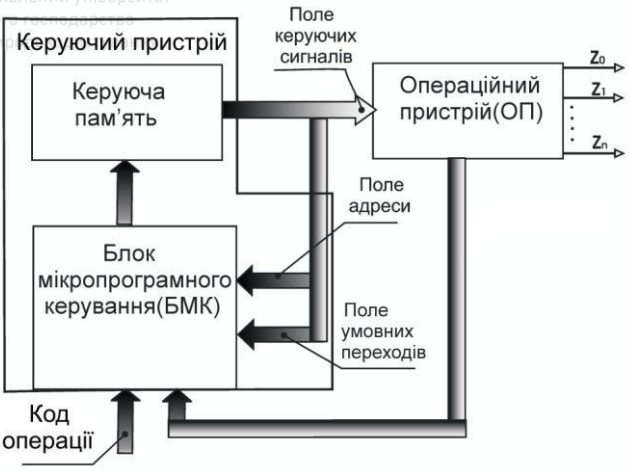


Рис. 11. Структурна схема мікропроцесора з керуючим пристроєм, побудованим за принципом програмованої логіки

Адреси наступних мікрокоманд знаходяться в полі адрес, яке передбачено у форматі мікрокоманди, що зчитується з керуючої пам'яті. Разом з ним передбачено поле переходів, один з розрядів якого вказує на безумовний перехід („0") або умовний перехід („1"). Для передачі керуючих сигналів в операційний пристрій використовується поле керуючих сигналів.

Таким чином, адреси мікрокоманд визначаються за допомогою блока мікропрограмного керування сигналами поля адрес і поля умовних переходів. Мікрокоманди послідовно зчитується з комірок керуючої пам'яті і забезпечують функціонування операційного пристрою на протязі виконання поточної мікропрограми, в кінці якої формується сигнал про її завершення і початок виконання наступної мікропрограми.

11. Мікросхемотехніка мікропроцесорних систем

11.1. Елементи пам'яті – тригери

Результат операції, яку виконує логічний елемент, визначається наявністю певних вхідних сигналів, характерних лише для цієї операції. Якщо вхідні сигнали змінюються, то змінюється також результат. У більшості випадків результат виконуваної логічної операції має зберігатися протягом певного часу. Щоб зберегти (запам'ятати)

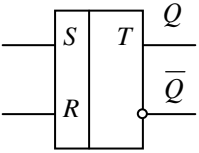


Рис. 12. Тригер з окремим запуском

результати окремих логічних операцій, треба мати спеціальні запам'ятовуючі пристрої, які, отримавши інформацію у вигляді логічної "1" чи "0", зберігатимуть її у незмінному вигляді доти, поки вона буде потрібна. Запам'ятовуючі елементи в пристроях мікропроцесорної техніки називають тригерами.

У загальному вигляді будь-який тригер можна розглядати як елемент, що має один або кілька керуючих входів і два виходи: прямий Q та інверсний $\overline{Q}$. Якщо на прямому виході тригера виникає напруга, що відповідає значенню логічної „1", то кажуть, що тригер перейшов у одиничний стан. Якщо на прямому виході тригера під дією вхідних сигналів напруга набуває значення, близького до нуля, то кажуть, що тригер перебуває в нульовому стані. Таким чином, тригер може перебувати в одному з двох стійких станів – одиничному або нульовому. Перехід з одного стану в інший здійснюється під час подання певних значень електричних сигналів на відповідні входи. При відсутності вхідних сигналів тригер зберігає свій останній стан. На відповідні входи тригерів подають інформаційні і керуючі сигнали. Інформаційними є сигнали, які треба запам'ятати. Керуючі – це ті сигнали, після подання яких інформація записується в тригер. За способом запису інформації розрізняють асинхронні і синхронні тригери. В асинхронних тригерах перехід у новий стан спричиняється змінами інформаційних вхідних сигналів. У синхронних тригерах такий перехід відбувається тільки після подання спеціальних керівних сигналів, які називають тактовими або синхронізуючими. Вхід тригера, на який надходять такі сигнали, має назву тактового і позначається літерою "*C*". За способом сприйняття тактових сигналів розрізняють тригери, які керуються рівнями напруг (статичне керування) і перепадами напруг (динамічне керування). Динамічний тактовий вхід може бути прямим або інверсним. У першому випадку тригер реагує на перепад тактового сигналу з нульового на одиничний, у другому – з одиничного в нульовий. Електричними сигналами для керування тригерами та логічними схемами є прямокутні імпульси, джерелом яких є генератор тактових імпульсів (ГТІ), стабілізований кварцовим резонатором. Найбільш поширеними є такі основні типи тригерів: з окремим запуском (*RS*-тригер), з інформаційним входом (*D*-тригер), лічильний (*TT*-тригер), та універсальний (*JK*-тригер).

Тригер з окремим запуском – "*RS*-тригер" (рис. 12) має два керуючі входи: вхід встановлення в одиничний стан – *S*-вхід (від set – встановити) і скидний вхід – *R*-вхід (від reset – скинути). Нехай у початковий момент часу тригер перебуває в нульовому стані, а напруга на його керуючих входах також відповідає логічному "0". Якщо на *S*-вхід тригера надійде перепад напруг від значення, яке дорівнює "0" до значення, яке дорівнює "1", то тригер перейде в одиничний стан. При цьому в тригер буде записана "1". Якщо тригер перейшов в одиничний стан, то будь-які електричні сигнали, подані на *S*-вхід, не призведуть до зміни цього стану, якщо на *R*-вході, як і раніше, напруга відповідатиме логічному "0". Таким чином тригер зберігає

"1". Щоб встановити *RS*-тригер в нульовий стан, якщо він перебуває в одиничному, треба подати перепад напруг з "0" в "1" на *R*-вхід і при цьому на *S*-вході підтримувати значення напруги, яке відповідає логічному "0"(табл.17).

Тригер з інформаційним входом – "*D*-тригер" (рис. 13) має два входи – інформаційний "*D*" і тактовий "*C*". Сигнал на виході *Q* інформаційного тригера повторює за значенням сигнал, поданий на інформаційний вхід "*D*", якщо на тактовому вході "*C*" встановлено рівень напруги, що дорівнює "1". Якщо тактового сигналу не буде, то незалежно від значення напруги на *D*-вході тригер зберігає свій останній встановлений стан. Отже, якщо на *D*-вході напруга дорівнює за значенням логічний одиниці, то, при надходженні "1" на *C*-вхід, на виході *Q* запишеться "1", а якщо *D*=0, то і на виході *Q* напруга також дорівнюватиме "0", якщо на тактовий вхід надійде "1".

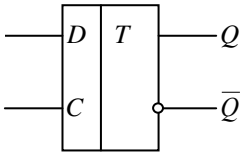


Рис. 13. Тригер з інформаційним входом

Лічильний тригер – *TT*-тригер (рис. 14) застосовують для побудови лічильників електричних імпульсів. Цей тип тригера має один керуючий (лічильний) вхід „*C"*. Лічильний тригер змінює свій стан на протилежний від кожного імпульсу, що надходить на вхід „*C*", а точніше – від перепаду напруги від "0" до "1" або від "1" до "0" (динамічне керування). Якщо *TT*-тригер перебуває в нульовому стані, то він буде переведений в одиничний стан першим імпульсом, який надходить на його вхід. Другий імпульс знову поверне його в нульовий стан і т. д. У проміжках між імпульсами тригер зберігає свій останній стан.

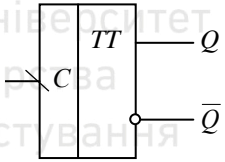


Рис. 14. Лічильний тригер

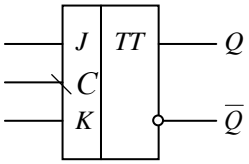


Рис. 15. Універсальний тригер

Універсальний тригер – *JK*-тригер (рис. 15) має два інформаційні входи "*J*" і "*K*" та один тактовий вхід. Він забезпечує такі режими роботи.

Якщо на входи "*J*" і "*K*" подати напругу, яка відповідає "1" (наприклад $+5B$), то тригер почне працювати в лічильному режимі. Роль лічильного входу в цьому разі виконує тактовий вхід "*C*". Такий тригер використовується як синхронний *RS*-тригер. При цьому його пуск здійснюється фронтами тактових імпульсів (при переході вхідної напруги від "1" до "0" або від "0" до "1"). Вхід "*J*" виконує функції входу "*S*", а вхід "*K*" відповідає входу "*R*". Тобто, якщо *J*=1 і *K*=0, то при надходженні на *C* негативного фронту імпульсу $\downarrow U_C$ тригер переходить в одиничний стан $Q=1$,

$\overline{Q}=0$. При J=0 і K=1 під час надходження негативного фронту тактового імпульсу відбувається скид тригера в нульове положення $Q=0$, $\overline{Q}=1$. Якщо обидва входи тригера мають низький потенціал J=0, K=0, то наявність тактових імпульсів на вході "C" не впливає на виходи тригера, який знаходиться в режимі очікування.

З простих логічних і запам'ятовуючих елементів будуються функціональні вузли.

11.2. Регістри

Регістри призначені для записування або тимчасового зберігання і перетворення двійкової інформації. Будь-який регістр складається із з'єднаних у певній послідовності запам'ятовуючих елементів – тригерів і керуючих логічних елементів. Кожний тригер призначений для введення, зберігання і виведення одного розряду двійкового числа. Загальна кількість тригерів у регістрі визначає його розрядність. Залежно від призначення всі регістри поділяють на паралельні, послідовні і перетворювальні.

Паралельні регістри призначені для введення, зберігання і виведення багаторозрядних двійкових чисел. Числа в такий регістр надходять одночасно всіма розрядами, тобто паралельним кодом (рис.16).

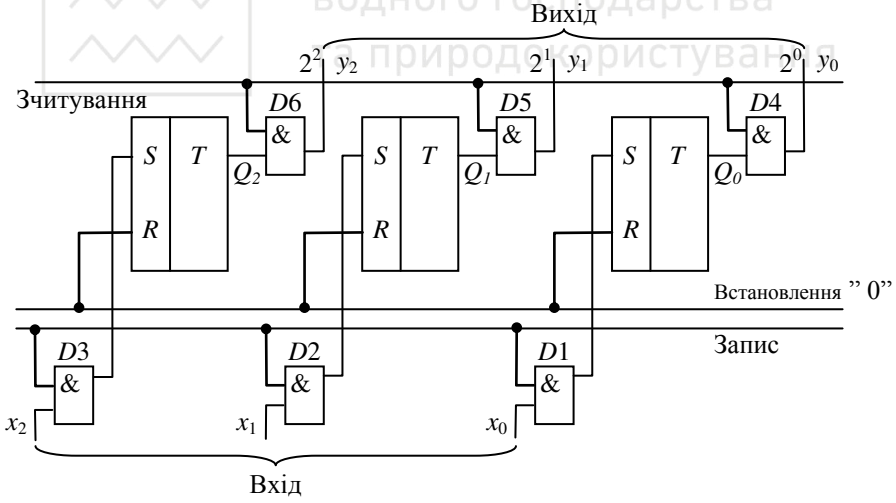


Рис. 16. Трирозрядний паралельний регістр

Двійкова інформація надходить на інформаційні входи. Перед записуванням цієї інформації всі тригери встановлюються в початковий нульовий стан. Для цього на керуючий вхід "встановлення 0", з'єднаний з R-входами тригерів,подається скидний позитивний імпульс, який дорівнює за

значенням “1”. Виходячи з принципу роботи *RS*-тригера ця дія призведе до встановлення нульового стану на прямих виходах Q_0, Q_1, Q_2. Під час подання позитивного імпульсу на керуючий вхід “запис” двійкове число через логічні елементи D_1, D_2, D_3, пропускається на *S*-входи тригерів. При цьому ті тригери, на *S*-входи яких надходить одиничний розряд двійкового числа, встановлюються в одиничний стан, а ті тригери, на *S*-входи яких надходять нульові розряди, будуть у нульовому стані. Таким чином, двійкова інформація з входів x_0, x_1, x_2 записується в тригери і зберігається там до надходження чергових сигналів на керуючі входи. Двійкова інформація з виходів тригерів Q_0, Q_1, Q_2 надходить на входи логічних елементів D_4, D_5, D_6 і після приходу імпульсу на керуючий вхід “зчитування” з’являється на виходах регістру Y_0, Y_1, Y_2. Паралельні регістри є регістрами пам’яті.

Послідовні регістри на відміну від паралельних мають тільки один інформаційний вхід „D” та один керуючий вхід синхронізації „C” . Їх використовують для введення, зберігання, зсуву і виведення двійкових чисел. В наведеній схемі чотирьохрозрядного послідовного регістра (рис.17) його старший розряд завдяки наявності інвертора на К-вході універсального тригера, працює в режимі D-тригера. Двійкові розряди надходять на інформаційний вхід „D” у послідовному коді, один за одним, через рівні проміжки часу. Запис починається з молодшого розряду двійкового числа, який подається на вхід регістра D. Після надходження синхронізуючого імпульсу на вхід „C” молодший розряд записується на виході тригера старшого розряду Q_3. Наступний розряд двійкового числа записується на виході Q_3 після приходу другого імпульсу, а двійковий розряд, що був раніше записаний на виході Q_3, переміститься на вихід Q_2 наступного тригера і.т.д. Отже з кожним наступним імпульсом раніше записана двійкова інформація зміщується на одну позицію вправо. Одночасно з інформаційними імпульсами на входи „C” надходять синхронізуючи імпульси, які забезпечують запис інформації в тригери. Після чотирьох тактів двійкове число, що подане на вхід, буде зафіксоване в регістрі.

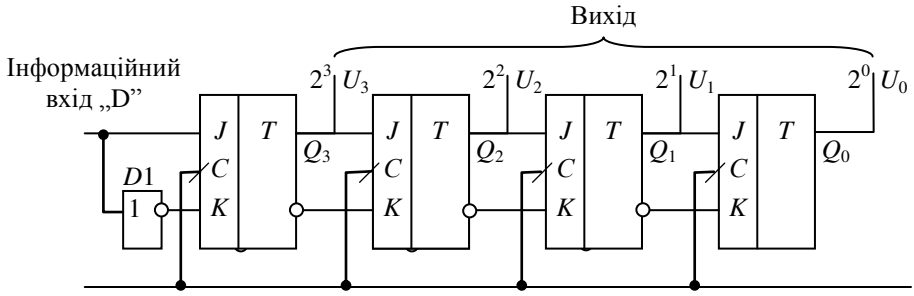


Рис. 17. Чотирирозрядний зсувний регістр

Послідовний регістр може виконувати також функції регістра зсуву, якщо в нього попередньо записати певне двійкове число, а на вхід „D” тригера старшого розряду подати логічний „0”.

Послідовний регістр може використовуватись для перетворення послідовного коду в паралельний. Існують реверсивні зсувні регістри, в яких направлення зсуву обирається в залежності від задач, що вирішуються. Слід зазначити, що зсув двійкового числа на один розряд вліво (у бік старших розрядів) відповідає його множенню на два, а вправо (у бік молодших розрядів) – діленню на два.

Перетворювальні регістри призначені для записування, зберігання, зсуву і перетворення двійкової інформації з паралельного коду в послідовний і навпаки. Вони є послідовно-паралельними і конструктивно виконуються як сукупність послідовного і паралельного регістрів.

11.3. Лічильники

Лічильники здійснюють підрахунок імпульсів, які надходять на їх входи, тимчасове зберігання кожного стану, перетворення послідовності імпульсів, що надходять на вхід лічильника, в паралельний двійковий код на його виходах, ділення частоти вхідного імпульсного сигналу. Лічильники можна поділити на чотири основні групи: прямої лічби (підсумовуючі), зворотньої лічби (віднімаючі), реверсивні (підсумовуючі і віднімаючі) і подільники частоти.

Розглянемо будову і принцип їх дії на прикладі трирозрядного двійкового підсумовуючого лічильника, виконаного на універсальних *JK*-тригерах (рис.18), та часові діаграми, які характеризують його роботу (рис.19). На входи *J* і *K* тригерів постійно подається логічна “1”, а вхідні імпульси $U_{вх}$ надходять на вхід синхронізації “*C*” першого тригера. Таким чином, як наводилося вище, універсальний тригер працює у лічильному режимі.

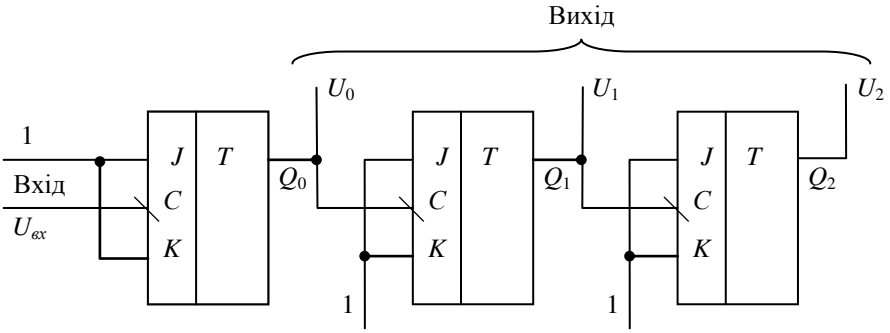


Рис.18. Схема трирозрядного двійкового лічильника

Лічильники прямої лічби працюють за принципом підсумовування імпульсів, що надходять на вхід. Загальна кількість імпульсів відображається на виводах лічильників у паралельному двійковому коді (рис.18). У початковий момент часу всі тригери знаходяться у нульовому стані. Після надходження на вхід регістра першого імпульсу $U_{вх}$=1, його заднім фронтом ($U_{вх}\downarrow$) в перший тригер записується "1" (Q_0=1). Якщо записати справа наліво стан усіх тригерів після першого вхідного імпульсу, то дістанемо таке двійкове число: $001_{(2)}=1_{(10)}$ (рис. 16). Другий імпульс своїм заднім фронтом переводить перший тригер в нульовий стан. При цьому напруга на виході Q_0 першого тригера зменшується з одиничного значення до нульового ($U_0\downarrow$), внаслідок чого "1" записується в другий тригер (Q_1=1). Таким чином, стан лічильника після другого імпульсу описується двійковим числом $010_{(2)}=2_{(10)}$. Третій імпульс встановлює перший тригер знову в одиничний стан і при цьому стан лічильника матиме вигляд: $011_{(2)}=3_{(10)}$. Після четвертого імпульса перший і другий тригери скидаються у нульовий стан, а логічна одиниця з'являється на виході третього тригера. Стан лічильника матиме вигляд $100_{(2)}=4_{(10)}$ Отже з кожним імпульсом вміст лічильника збільшується на „1". Після 8 імпульсів лічильник скидається у нульовий стан і далі процес лічби поновлюється.

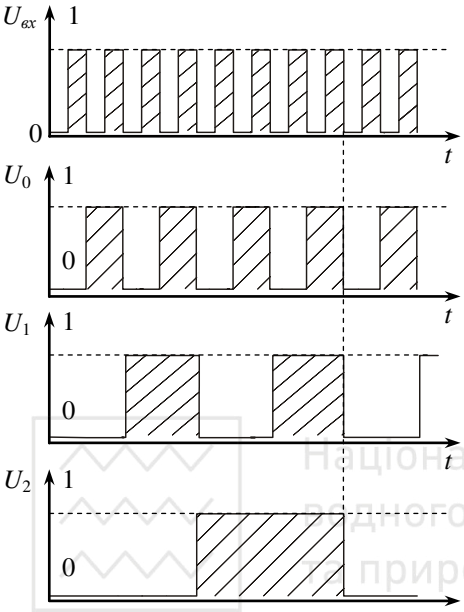


Рис. 19. Часові діаграми трирозрядного двійкового лічильника

11.4. Шифратори

В пристроях мікропроцесорної техніки інформація подається у вигляді кодованих двійкових сигналів. Функціональні вузли, які виконують функції кодування, мають назву шифраторів. Існують різноманітні схеми шифраторів. Розглянемо принцип їх побудови на прикладі схеми шифратора, який перетворює десяткові числа на двійкові (рис. 20а). Шифратор складається з логічних елементів АБО (*DD*1,…,*DD*4), з'єднаних між собою згідно логіки, що відображає принцип його дії. Схема шифратора має входи,

які відповідають десятковим числам від “0” до ”9”, та чотири виходи, кожний з яких відповідає певному розряду двійкового числа – від нульового до третього. Наявності певного сигналу на вході і логічних “1” на виходах шифратора відповідають високі рівні потенціалів, а відсутності сигналів на входах і логічних “0” на його виходах – низькі рівні потенціалів. Таблиця станів шифратора (табл. 18) має такий вигляд:

Таблиця 18

Десяткові числа	Двійкові числа			
	2^3(D)	2^2(C)	2^1(B)	2^0(A)
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

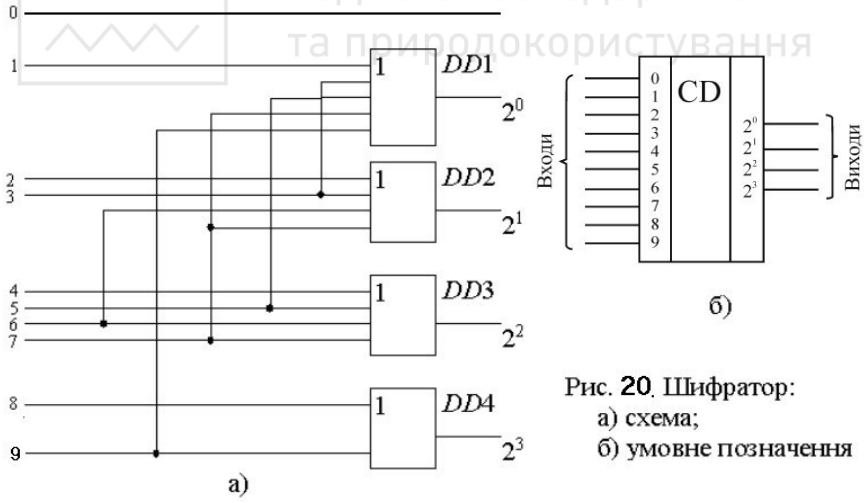


Рис. 20. Шифратор:
а) схема;
б) умовне позначення

11.5. Дешифратори

Дешифратори — це комбінаційні цифрові автомати з *n* входами та N виходами, які перетворюють вхідні двійкові сигнали в сигнали високого

рівня ("1") на одному з виходів, десятковий код якого відповідає вхідному коду: $y(j) = \begin{cases} 1 & при \quad x = j \\ 0 & при \quad x \neq j \end{cases}$, де j=0, 1, 2, 3, … 2^{n-1}.

Розрізняють повні ($N = 2^n$) і неповні ($N<2^n$) дешифратори. Розглянемо побудову найпростішого лінійного дешифратора з двома входами n = 2 на логічних елементах І, НІ. Враховуючи, що кількість виходів $N = 2^2 = 4$, таблиця його станів має такий вигляд:

Таблиця 19

Входи		*Виходи*			
x_1	x_0	y_0	y_1	y_2	y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

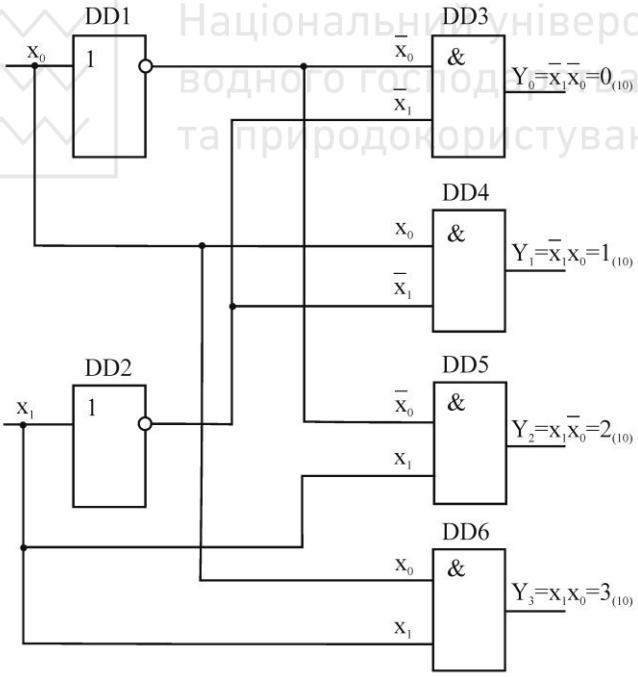


Рис.21. Лінійний дешифратор з двома входами

Таблиці станів відповідають наступні логічні рівняння:

$y_0 = \bar{x}_1 \cdot \bar{x}_0;$ $\quad y_1 = \bar{x}_1 \cdot x_0;$

$y_2 = x_1 \cdot \bar{x}_0;$ $\quad y_2 = x_1 \cdot x_0;$

Виходячи з них, будуємо структурну схему дешифратора на два входи та чотири виходи з простих логічних елементів НІ (DD1, DD2) та І (DD3, DD4, DD5, DD6) (рис. 21). В десятковій системі вихідні коди, що відповідають двійковим кодам, дорівнюють:

$y_{0\,10} = 0 \cdot 2^1 + 0 \cdot 2^0 = 0_{10}$; $\quad y_{2\,10} = 1 \cdot 2^1 + 0 \cdot 2^0 = 2_{10}$;

$y_{1\,10} = 0 \cdot 2^1 + 1 \cdot 2^0 = 1_{10}$; $\quad y_{3\,10} = 1 \cdot 2^1 + 1 \cdot 2^0 = 3_{10}$;

До складу неповного чотирирозрядного дешифратора $N=4<2^4$ (рис.22а) входять чотири логічних елемента "НІ" (*DD*1,…,*DD*4), та чотири логічних елемента "І" (*DD*5,…,*DD*8).

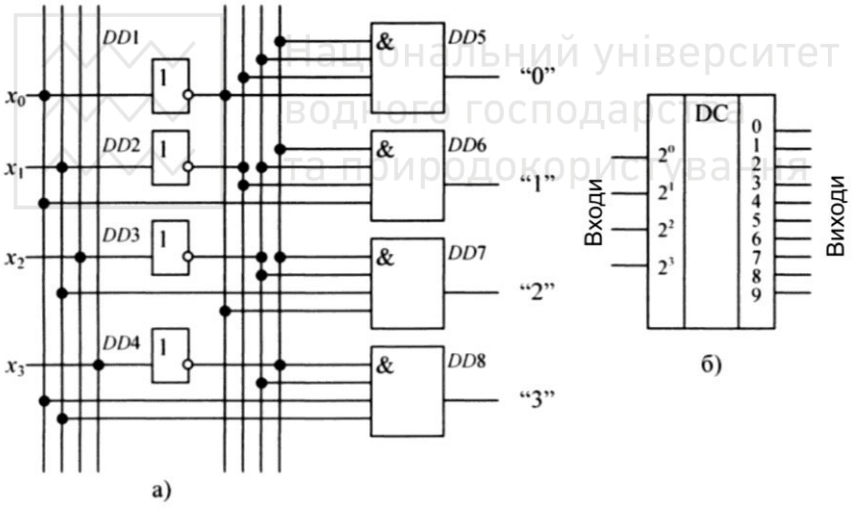


Рис.22. Чотирирозрядний дешифратор: а) схема; б) умовне позначення

Якщо, наприклад, на входи дешифратора надходить двійкова комбінація 0100 (x_0=0, x_1=1, x_2=0, x_3=0), то на всі чотири входи логічного елемента *DD*7 надходять логічні одиниці і тому на його виході, що відповідає десятковому числу "2", також з'являється логічна одиниця. Всі інші виходи

дешифратора при цьому мають нульовий потенціал. Умовне позначення дешифратора наведене на рис. 22б.

11.6. Суматори

Суматори – це функціональні вузли, що призначені для арифметичного додавання двійкових чисел. Розрізняють комбінаційні суматори, побудовані з логічних елементів, та нагромаджувальні, в основі яких лежать тригери. Найбільш розповсюдженими є комбінаційні суматори.

Схема для підсумовування двох однорозрядних двійкових чисел має назву півсуматора. Вона побудована на логічних елементах "НІ": $D1$, $D2$, логічних елементах "І": $D3$, $D4$ та логічному елементі $D5$ – "АБО". Доданки подаються на входи півсуматора x_1; x_2, виходом суми якого є "S". Щоб врахувати перенесення до старшого розряду, в схему додатково вводять ще один логічний елемент „I" — $D6$, на виході якого з'являється "1" тоді, коли обидва вхідних сигнали також рівні "1". Цей вихід має назву "значення перенесення" – p. Таблиця істинності схеми півсуматора має такий вигляд:

Таблиця 20

x_1	x_2	S	p
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Отже, ця схема підсумовує два однорозрядні двійкові числа і має два входи, на які подаються розряди доданків і два виходи: суми S і перенесення p. Окремо півсуматор випускають в мікросхемному виконанні і його умовне позначення зображено на рис. 23б.

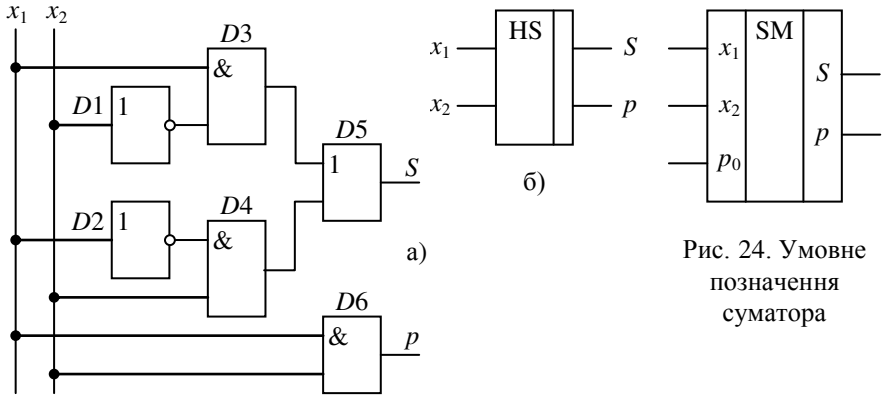


Рис. 24. Умовне позначення суматора

Рис. 23. Півсуматор: а) схема; б) умовне позначення

Однорозрядний повний суматор відрізняється від півсуматора тим, що він призначений для додавання трьох однорозрядних двійкових чисел: розряду першого доданка x_1, розряду другого доданка x_2 і перенесення з попереднього розряду p. Логіка роботи повного суматора описується таблицею 21.

Таблиця 21

x_1	x_2	p_0	S	p_1^*
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

p_1^* – перенесення до наступного розряду.

Однорозрядний повний суматор будують на базі півсуматорів. Функціональне позначення повного однорозрядного суматора, що випускається в мікросхемному виконанні, наведено на рис. 24. Для підсумовування багаторозрядних двійкових чисел треба з'єднати однорозрядні суматори так, щоб відбувалось перенесення з молодших розрядів в старші розряди.

11.7. Мультиплексори і демультиплексори

Мультиплексори по черзі комутують сигнали кількох джерел інформації: $x_0;x_1;x_2;\ldots;x_{n-1}$ на спільний вихід D. Номер вхідної лінії, що під'єднується до виходу на кожному такті машинного часу, визначається адресним кодом $A_0;A_1;A_2;\ldots;A_{m-1}$. Таким чином, мультиплексор забезпечує передачу даних від «n» джерел послідовно в часі: $n=2^m$ (при наявності «m» адресних входів).

Найбільш розповсюдженими є мультиплексори з внутрішнім дешифратором DC, які містять також логічні схеми «І», кількість яких відповідає кількості інформаційних входів, та вихідну логічну схему «АБО». Будова мультиплексора з чотирма інформаційними і двома адресними входами наведена на схемах рис.25.

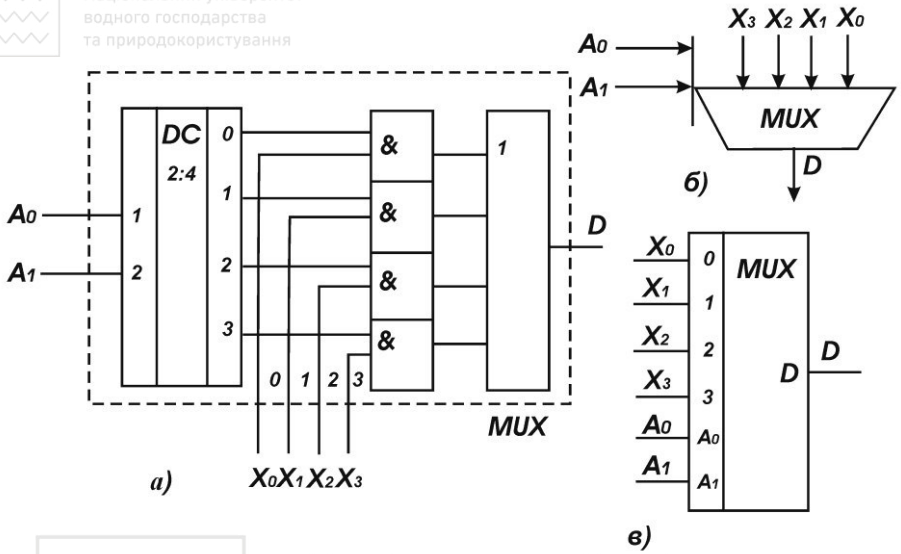


Рис.25 Мультиплексор: а) схема; б) умовне позначення на функціональних схемах; в) умовне позначення на принципових схемах.

Сигнали $x_0;x_1;x_2;x_3$ надходять на інформаційні входи 0;1;2;3 мультиплексора MUX, а на його адресні входи $A_0;A_1$ з лічильника мікроконтролера надходить двійковий код, що визначає, який з входів MUX буде з'єднаний з його виходом в даний момент часу.

Разом зі схемами з внутрішніми дешифраторами використовують схеми мультиплексорів з адресними мінтермами.

Демультиплексор – це функціональний вузол мікропроцесорної техніки, призначений для комутації сигналу з одного інформаційного входу «D» на один з «n» інформаційних виходів. Адреса виходу, на який в кожний такт машинного часу передається вхідний сигнал, визначається її кодом $A_0;A_1;A_2;\ldots;A_{m-1}$. Адресні входи «m» та інформаційні виходи «n» зв'язані між собою співвідношенням: $m=\log_2^n$. Таким чином функція демультиплексора є зворотною функції мультиплексора. Розглянемо його будову і принцип дії на прикладі схеми з внутрішнім дешифратором і чотирма виходами (рис.26).

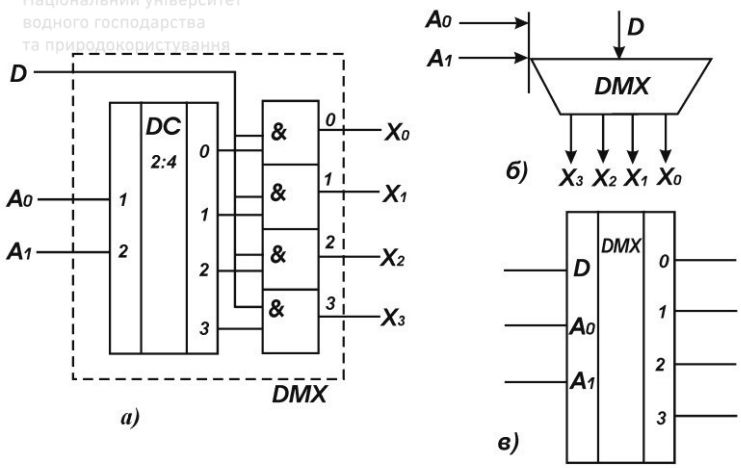


Рис.26 Демультиплексор: а) схема; б) умовне позначення на функціональних схемах; в) умовне позначення на принципових схемах.

При наявності вхідного сигналу «D» та логічної «1» на одному з виходів внутрішнього дешифратора DC, на виході відповідної логічної схеми «І» також з'являється певний сигнал.

Мультиплексори MUX та демультиплексори DMX називають також *селекторами даних*. Мультиплексори використовують для комутації окремих інформаційних ліній і шин та перетворення паралельного коду у послідовний, а демультиплексори – для комутації окремих ліній і шин та перетворення послідовного коду у паралельний. Вони також використовуються для реалізації логічних функцій.

11.8. Цифро-аналогові перетворювачі (ЦАП)

Ці пристрої перетворюють інформацію з цифрової двійкової форми в аналогову. Одним з можливих варіантів ЦАПа є схема з двійковозваженими опорами, яка виконується на базі операційного підсумовуючого підсилювача ОП. Його вихідна напруга є сумою «n» вхідних сигналів, кожний з яких відповідає певному розряду двійкового коду. Розглянемо принципову схему ЦАПа з двійково-зваженими опорами (рис.27). Кожний n-ий розряд визначається станом транзисторних ключів $К_0$, $К_1$, $К_2$…$К_{n-1}$, які під'єднують до входу операційного підсилювача резистори, що з'єднані з джерелом

опорної напруги $U_{оп}$. Якщо опір резистора молодшого нульового розряду має значення R_0, то опори наступних старших розрядів визначаються із співвідношення: $R_{n-1} = R_0 / 2^{n-1}$. Таким чином при n=1, $R_0 = R_0 / 1$; n=2, $R_0 = R_0 / 2$; n=3, $R_2 = R_0 / 4$; n=4, $R_3 = R_0 / 8$ і т.д.

На вході операційного підсилювача ОП завжди виникає практично нульовий потенціал і тому його вхідний струм визначається співвідношенням:

$$I_к = \frac{2^{n-1} \cdot K_{n-1}}{R_0} \cdot U_{оп} + \frac{2^{n-2} \cdot K_{n-2}}{R_0} \cdot U_{оп} + ... + \frac{2^1 \cdot K_1}{R_0} \cdot U_{оп} + \frac{2^0 \cdot K_0}{R_0} \cdot U_{оп} =$$

$$\frac{U_{оп}}{R_0} \cdot \left(K_{n-1} \cdot 2^{n-1} + K_{n-2} \cdot 2^{n-2} + ... + K_1 \cdot 2^1 + K_0 \cdot 2^0 \right) = \frac{U_{оп}}{R_0} \sum_{i=0}^{n-1} 2^i \cdot K_i,$$

де K_{n-1}; K_{n-2};…K_1;K_0 – відповідні стани транзисторних ключів («1» - відкритий, «0» - закритий).

При цьому вихідна напруга ЦАП визначається за формулою:

$$U_{вих} = -I_к \cdot R_{зз} = -\frac{U_{оп}}{2^n} \sum_{i=0}^{n-1} 2^i \cdot K_i = -MN_{(10)}$$

де $R_{зз} = R_0 / 2^n$ - опір в колі зворотного зв'язку оперативного підсилювача; $M = U_{оп} / 2^n$ - масштабний коефіцієнт; $N_{(10)}$ - десяткове значення двійкового коду, введеного в регістр.

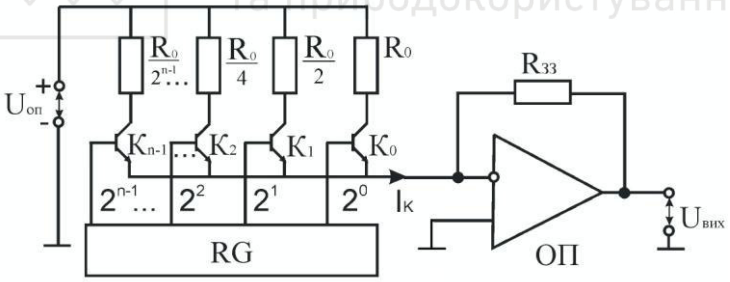


Рис. 27 Принципова схема ЦАП з двійково-зваженими опорами:

RG – паралельний регістр; R_0 , $R_0 / 2$, $R_0 / 4$, …, $R_0 / 2^{n-1}$ - опори; ОП – операційний підсилювач; K_0, K_1, K_2, …, K_{n-1} – транзисторні ключі.

Розглянемо обчислення вихідної напруги 8 – розрядного ЦАПа з $U_{оп}$= 5В. Якщо наприклад, двійкове число дорівнює $10111010_{(2)}$, то $N_{(10)}=186_{(10)}$; M = 0,0195, а вихідна напруга. $|U_{вих}|=0{,}0195 \cdot 186 \approx 3{,}63$В

Недоліком розглянутого типу ЦАП є жорсткі вимоги до стабільності і точності номінальних опорів, резисторів. При зміні їх опорів змінюється також значення вагових коефіцієнтів і зростає похибка перетворення.

Для усунення цього недоліку використовують схему ЦАПа з резисторною матрицею R-2R (рис.28а). До його складу входить матриця, яка містить резистори тільки двох номіналів R і 2R, операційний підсилювач ОП та перемикачі на транзисторних ключах VT (рис.28б), які показані на схемі як гіпотетичні механічні перемикачі $К_0$, $К_1$,…, $К_3$ (рис.28а).

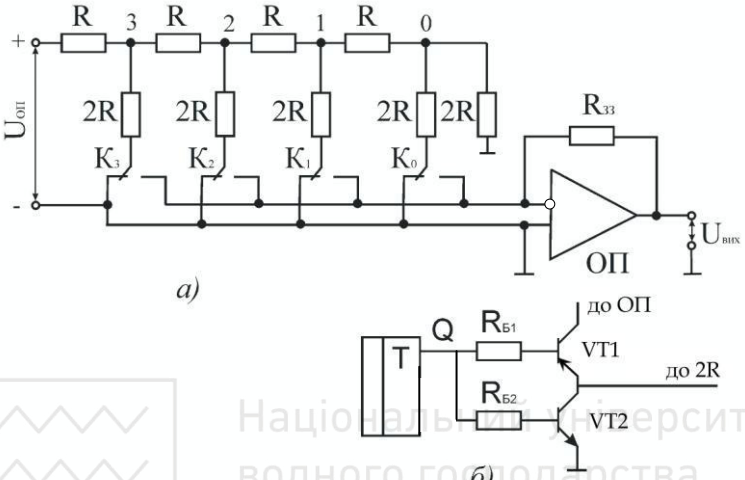


Рис.28а. Принципова схема ЦАП з резисторною матрицею: R - 2R – матриця; ОП – операційний підсилювач; $К_0$ – $К_3$ – перемикачі; R_{33} – опір зворотного зв'язку. Рис.28б. Перемикач на транзисторних ключах

Особливість резисторної матриці полягає в тому, що її опори у вузлах «0», «1», «2», «3» відносно нульового потенціалу корпуса («маси») із врахуванням «правих» (2R) та «верхніх» (R) резисторів (рис.28а) є рівними:

$$R_0 = \frac{2R}{2} = R,\ R_1 = \frac{R_0 + R \,\|\, 2R}{R_0 + R \,\|\, 2R} = R,\ R_2 = \frac{R_1 + R \,\|\, 2R}{R_1 + R \,\|\, 2R} = R,\ R_3 = \frac{R_2 + R \,\|\, 2R}{R_2 + R \,\|\, 2R} = R.$$

Ці опори створюють у вузлах подільники напруги на два і тому:

$$U_0 = \frac{U_1}{2};\ U_1 = \frac{U_2}{2};\ U_2 = \frac{U_3}{2};\ U_3 = \frac{U_{оп}}{2}.$$

Тобто, якщо на вхід матриці подається опорна напруга $U_{оп}$, то в її вузлах «3», «2», «1», «0» утворюються відповідні напруги: $U_3 = 1/2 \cdot U_{оп}$; $U_2 = 1/4 \cdot U_{оп}$; $U_1 = 1/8 \cdot U_{оп}$; $U_0 = 1/16 \cdot U_{оп}$ і коефіцієнт передачі між сусідніми вузлами матриці

$$k_{ПВ} = \frac{U_0}{U_1} = \frac{U_1}{U_2} = \frac{U_2}{U_3} = 0{,}5.$$

Разом з цим, якщо розглянути, наприклад, потенціал вузла «3» матриці, коли він підключений до входу операційного підсилювача ($К_3$=1), то

його величина буде дорівнювати $U_з = U_{оп} \cdot \frac{R}{R+2R} = \frac{U_{оп}}{3}$ (рис.28а). Враховуючи, що коефіцієнт передачі між вузлами $k_{пв}$=0,5, отримаємо:

- для вузла «2» при $К_2$=1: $U_2 = 1/2 \cdot U_з = 1/2 \cdot U_{оп}/3$;
- для вузла «1» при $К_1$=1: $U_1 = 1/2 \cdot U_2 = 1/4 \cdot U_{оп}/3$;
- для вузла «0» при $К_0$=1: $U_0 = 1/2 \cdot U_1 = 1/8 \cdot U_{оп}/3$.

Підключення вузлів матриці R-2R до операційного підсилювача ОП здійснюється за допомогою транзисторних ключів VT1, VT2, що керуються двійковим кодом з тригерів «Т» паралельних регістрів або лічильників (рис.28б).

При цьому електричні сигнали з певних вузлів надходять на вхід операційного підсумовуючого підсилювача через окремі резистори 2R і передаються на його вихід з коефіцієнтом передачі $k = \frac{R_{зз}}{2R}$. Вихідна напруга «n» - розрядного ЦАПа визначається як сума окремих сигналів за формулою:

$$U_{вих} = -\frac{1}{3} \cdot \frac{U_{оп}}{2^n} \cdot \frac{R_{зз}}{R} \cdot \sum_{i=0}^{n-1} 2^i \cdot K_i = -MN_{(10)}$$

де $M = \frac{1}{3} \cdot \frac{U_{оп}}{2^n} \cdot \frac{R_{зз}}{R}$ - масштабний коефіцієнт, величина якого визначається співвідношенням $R_{зз}/R$ (наприклад при $R_{зз}$=3R: $M = U_{оп}/2^n$); $N_{(10)}$ – десяткове значення двійкового коду, введеного в регістр; $К_i$ – стани транзисторних ключів («0» або «1»).

11.9. Аналого-цифрові перетворювачі (АЦП)

Для аналого-цифрового перетворення неперервний сигнал квантується і його значення вимірюються через рівні проміжки часу, причому, кожна дискрета представляється в двійковому коді. Частота квантування повинна дорівнювати подвійній максимальній частоті аналогового сигналу $f_{кв} = 2f_{\max}$. У випадках вимірювань напруг сигналів, що змінюються повільно і коли достатньою є швидкість квантування 2-2,5 вимірів/секунду, - використовують АЦП двотактного інтегрування (рис. 29).

Цикл роботи перетворювача $t_ц$ складається з двох тактів t_1 і t_2. У

першому такті на протязі фіксованого часу t_1 відбувається інтегрування вхідного сигналу $U_{вх}$, який через комутатор $DD1$ надходить на вхід.

Зростання вихідної напруги інтегратора відбувається за законом :

$U_{IT} = \frac{1}{RC}\int_0^{t_1} U_{вх}\,dt$ (рис. 30) і в момент часу t_1 напруга стає рівною $U_1 = U_{IT1}$ при $U_{вх1}$ (або $U_1 = U_{IT2}$ при $U_{вх2}$).

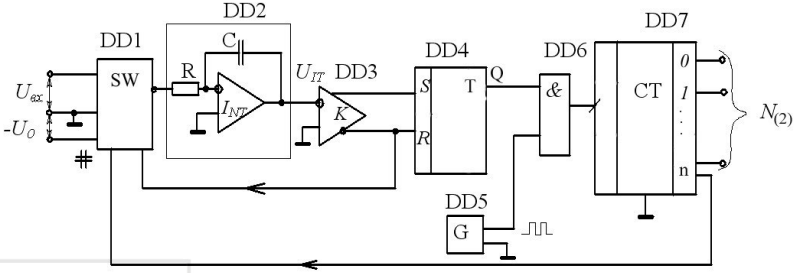


Рис.29. Схема АЦП двотактного інтегрування: $DD1$ — електронний комутатор; $DD2$ — інтегратор; $DD3$ — компаратор; $DD4$ — тригер; $DD5$ — генератор тактових імпульсів; $DD6$ — логічна схема „І”; $DD7$ — двійковий лічильник.

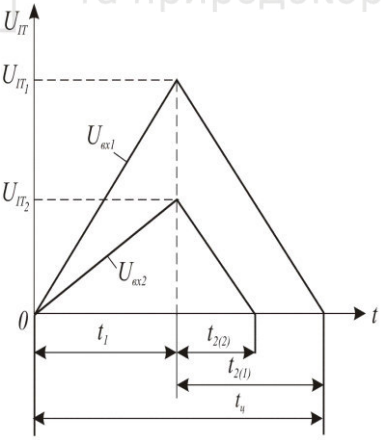


Рис. 30. Часова діаграма інтегратора АЦП:
U_{IT} – вихідна напруга; $U_{вх1}, U_{вх2}$ – вхідні напруги;
t_1 =const – час інтегрування; $t_{2(1)}$, $t_{2(2)}$ – час зчитування.

Ця напруга подається на один з входів компаратора $DD3$, другий вхід якого з’єднаний з нульовим потенціалом. Компаратор спрацьовує і на його

прямому виході з'являється логічна одиниця „1", яка встановлює в одиничний стан тригер $DD4$. Завдяки цьому логічна схема „І" $DD6$, один з входів якої підключений до прямого виходу тригера $DD4$, а другий — до генератора тактових імпульсів $DD5$, — періодично відкривається його імпульсами. З виходу $DD6$ імпульси надходять на вхід двійкового лічильника $DD7$. Тривалість першого такту t_1 визначається їх заданою кількістю $N_{\max} = 2^n$ (n-кількість розрядів лічильника). Після надходження останнього імпульсу відбувається переповнення лічильника і він скидається у нульовий стан, а комутатор $DD1$ переключає інтегратор $DD2$ з вхідної напруги $U_{вх}$ на опорну напругу протилежної полярності „$-U_0$".

Починається другий такт, при якому відбувається зчитування тактових імпульсів генератора $DD5$ на протязі змінного часу t_2. При цьому вихідна напруга інтегратора зменшується за законом :

$$U_{ІТ} = U_{1} - \frac{1}{RC}\int_0^{t_2} U_0 \, dt$$

і в момент часу t_2 стає рівною нулю (рис. 30). Кількість імпульсів, що зчитується і фіксується двійковим лічильником за час t_2, є пропорційною вхідній напрузі $U_{вх}$ (по середньому значенню):

$$N = U_{вх} \frac{N_{\max}}{U_0} = U_{вх} \frac{2^n}{U_0} = kU_{вх}.$$

В момент часу $t = t_2$ отримане двійкове число дешифрується і виводиться на 7-сегментний індикатор. При цьому вихідна напруга інтегратора стає рівною нулю ($U_{ІТ} = 0$) і компаратор також повертається у нульовий стан. Логічна „1", що з'являється на його інверсному виході, надходить на скидний вхід тригера $DD4$ та комутатор $DD1$. Тригер блокує надходження імпульсів на лічильник, а комутатор підключає інтегратор до вхідної напруги $U_{вх}$ і далі процес перетворення повторюється.

АЦП двотактного інтегрування забезпечують високу точність перетворення, але мають недостатню швидкодію, яка визначається величиною вхідної напруги $U_{вх}$.

Значно більшу швидкодію перетворень забезпечує АЦП послідовного наближення (рис. 31).

Його принцип дії ґрунтується на тому, що вихідний двійковий сигнал N повторно перетворюється ЦАП в аналоговий сигнал $U_{ЦАП}$ і порівнюється з вхідною аналоговою напругою $U_{вх}$. На виході компаратора $DD1$ з'являється логічна „1", якщо $U_{вх} > U_{ЦАП}$, або логічний „0", якщо $U_{вх} < U_{ЦАП}$. Імпульсний пусковий сигнал $U_{ПУСК}$ вмикає генератор тактових

імпульсів $DD2$ і одночасно записує логічну „1” в старший розряд „n-1” зсувного регістру послідовного наближення $DD3$. Інші його розряди при цьому скидаються у нульовий стан.

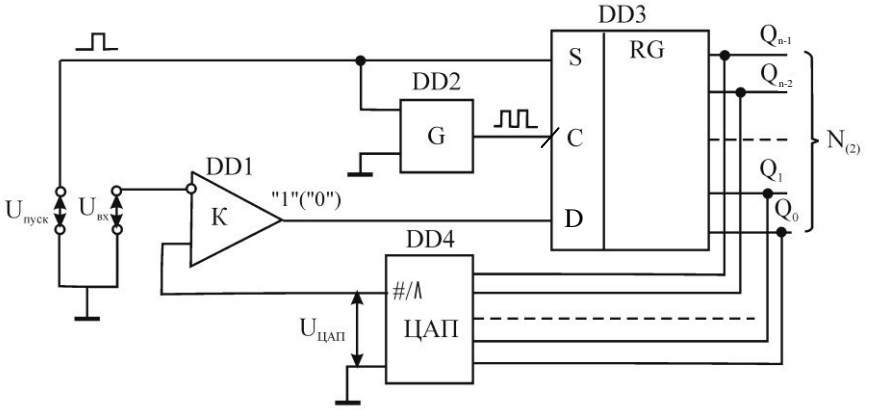


Рис. 31. Структурна схема АЦП послідовного наближення: $DD1$ — компаратор; $DD2$ — генератор тактових імпульсів; $DD3$ — зсувний регістр послідовного наближення; $DD4$ — цифро-аналоговий перетворювач (ЦАП).

З надходженням першого тактового імпульсу компаратор $DD1$ порівнює вхідну напругу $U_{вх}$ і вихідну напругу $DD4$ $U_{ЦАП}$, яка відповідає значенню старшого розряду $DD3$. Якщо при цьому $U_{ВХ} > U_{ЦАП}$, то на виході компаратора з’являється „1” і при наступному тактовому імпульсі зсувний регістр $DD3$ встановлює в „1” наступний найбільш значимий розряд „n-2” і т.д.

Коли вихідна напруга ЦАП стане більшою за вхідну напругу: $U_{ЦАП} > U_{ВХ}$, то на виході компаратора з’явиться логічний „0”.При наступному тактовому імпульсі „1” в поточному розряді зсувного регістра буде замінена на „0”, а в сусідньому молодшому розряді буде встановлена „1”. Таким чином процес перетворення є послідовністю апроксимацій вхідної напруги $U_{вх}$. Після останньої апроксимації цифровий двійковий сигнал перетворювача $N_{(2)}$ повністю відповідає вхідній напрузі $U_{вх}$ і подається на дешифратор індикатора або в порти мікропроцесорної системи. Цикл перетворення відбувається за час $t_n = n \cdot T + 1$, де n – розрядність регістра $DD3$; T – період тактових імпульсів генератора $DD2$. Далі процес перетворення повторюється.

Розглянемо процес перетворення у двійковий код вхідної напруги $U_{вх} = 1{,}76B$ восьмирозрядним АЦП (рис. 31). При надходженні 1-го тактового імпульсу найбільш значимий 7-ий біт регістра встановлений в „1” і двійкове число на виході АЦП дорівнює: $N_{(2)} = 10000000_{(2)}$. Вихідна напруга ЦАП: $U_{ЦАП} = \left| M \times N_{(10)} \right|$, де $M = \frac{1}{100}$ - масштабний коефіцієнт; $N_{(10)}$ – десяткове значення двійкового числа. Тому $U_{ЦАП} = \frac{2^7 \times 1}{100} = 1.28$ В. При цьому вхідна напруга залишається більшою, ніж вихідна напруга ЦАП: $U_{вх} = 1{,}76B > U_{ЦАП} = 1{,}28B$ і на виході компаратора залишається „1”. Тому наступний імпульс встановлює в „1” 6-ий біт регістру: $N_{(2)} = 11000000_{(2)}$;

$U_{ЦАП} = \frac{2^7 \times 1 + 2^6 \times 1}{100} = 1{,}92B > U_{вх} = 1{,}76B$. На виході компаратора з’являється „0” і тому черговий імпульс записує в 6-ий розряд „0”, а в наступний 5-ий — „1”. Вихідний цифровий код стає рівним $N_{(2)} = 10100000_{(2)}$;

$$U_{ЦАП} = \frac{2^7 \times 1 + 2^6 \times 0 + 2^5 \times 1}{100} = 1{,}6B < U_{вх} = 1{,}76B.$$

На виході компаратора знову з’являється „1” і наступний імпульс записує „1” в четвертий розряд регістру: $N_{(2)} = 10110000_{(2)}$;

$$U_{ЦАП} = \frac{2^7 \times 1 + 2^6 \times 0 + 2^5 \times 1 + 2^4 \times 1}{100} = 1{,}76B = U_{вх}!$$ На цьому процес перетворення закінчується (рис. 32). Отриманий код надходить в порти мікроконтролера, або на дешифратор та індикатор.

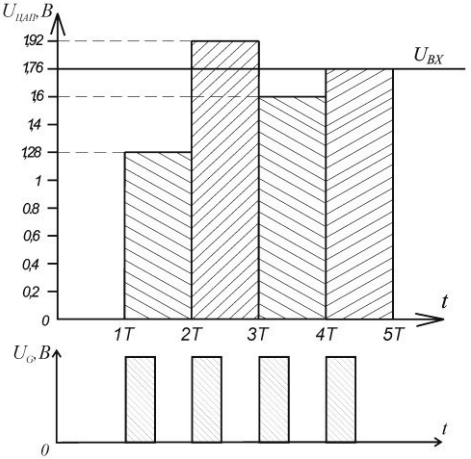


Рис. 32. Часова діаграма роботи АЦП послідовного наближення.

12. Приклади побудови цифрових пристроїв

12.1. Побудова комбінаційних цифрових пристроїв

Як зазначено вище, процес розробки цифрових пристроїв розпочинають з логічного синтезу, результатом якого є складання логічних рівнянь пристрою. Будь-яке логічне рівняння комбінаційного цифрового пристрою містить певну кількість операцій І, АБО, НІ. Кожна з них реалізується за допомогою відповідного логічного елемента. З'єднавши їх у відповідну послідовність, задану рівнянням, отримують структурну схему пристрою. Отже будь-який цифровий комбінаційний пристрій можна побудувати з комплектів трьох логічних елементів І, АБО, НІ.

Наприклад, побудуємо схему, яка реалізує логічне рівняння $y = x_1 \cdot \overline{x}_3 + x_2 \cdot \overline{x}_3 + \overline{x}_1 \cdot x_3 + \overline{x}_1 \cdot x_2$ на елементах І, АБО, НІ (рис. 33) :

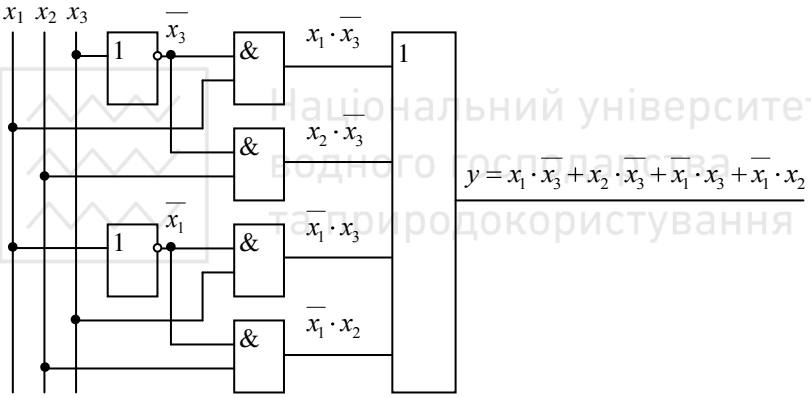


Рис. 33. Побудова функціональної схеми на елементах І, АБО, НІ

Але промислові інтегральні мікросхеми будь-якого функціонального призначення будують в основному на базі логічних елементів І-НІ або АБО-НІ. Щоб реалізувати логічне рівняння на цих елементах, його попередньо треба перетворити таким чином, щоб у ньому не було відповідно знаків логічного додавання або множення (використавши закони де Моргана). Побудуємо схему, яка реалізує логічне рівняння $y = x_1 \cdot \overline{x_2} + \overline{x_1} \cdot x_2$ на елементах І-НІ:

$$y = x_1 \cdot \overline{x_2} + \overline{x_1} \cdot x_2 = \overline{\overline{x_1 \cdot \overline{x_2}} + \overline{\overline{x_1} \cdot x_2}}\,.$$

Оскільки в рівнянні є знак логічного додавання, то перетворимо вираз під верхнім знаком інверсії за законом де Моргана. Дістанемо:

$y = x_1 \cdot x_2 + x_1 \cdot x_2 = x_1 \cdot x_2 \cdot x_1 \cdot x_2$ і після чого будуємо функціональну схему (рис. 34):

Щоб задане логічне рівняння реалізувати на елементах АБО-НІ, його перетворюють так, щоб у ньому не було знаків логічного множення.

Побудуємо схему, що реалізує логічне рівняння $y = x_1 \cdot \overline{x_2} + \overline{x_1} \cdot x_2$ з

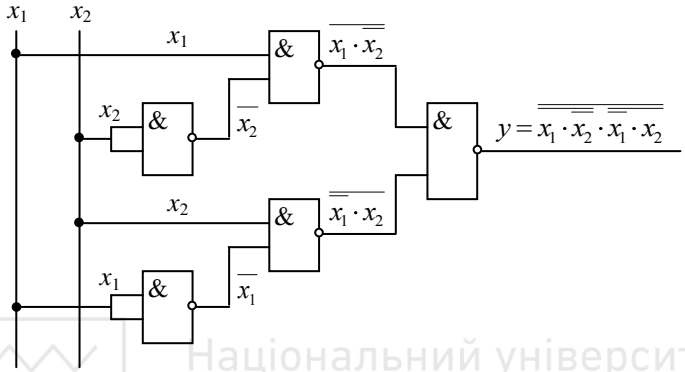


Рис. 34. Побудова функціональної схеми на елементах І-НІ

використанням логічних елементів типу АБО-НІ. Після перетворень де Моргана отримуємо вираз:

$$y = \overline{x_1 \cdot x_2} + \overline{x_1 \cdot x_2} = \overline{x_1} + x_2 + x_1 + \overline{x_2}$$

і будуємо схему, яка має вигляд (рис. 35):

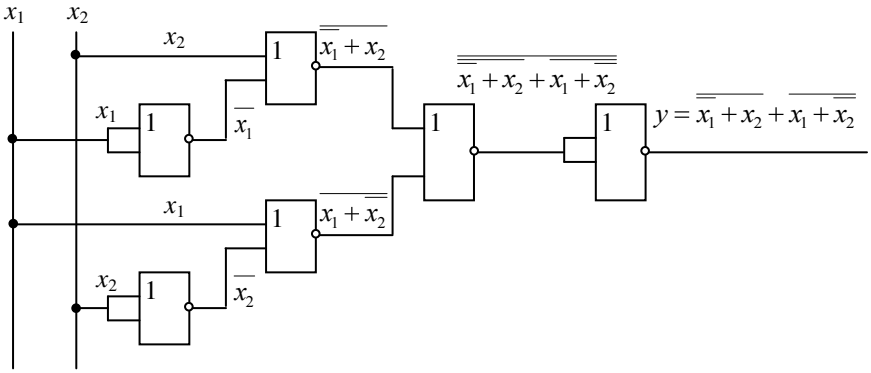


Рис. 35. Побудова функціональної схеми на елементах АБО-НІ

12.2. Побудова операційного пристрою

Розглянемо методику побудови операційних пристроїв на прикладі реалізації операції множення двійкових чисел.

При виконанні множення формуються часткові добутки (множеного на окремі розряди множника), які додаються при відповідних зсувах відносно одне одного.

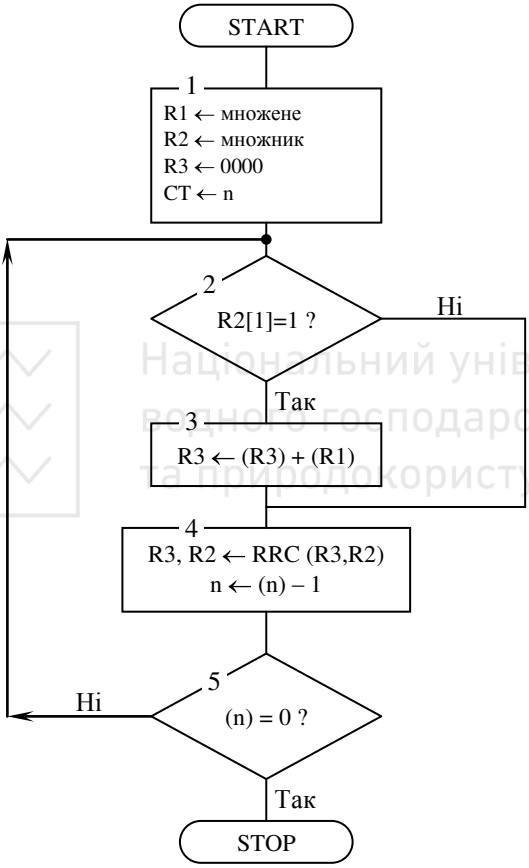


Рис. 36. Структурна схема алгоритму операції множення

Тому при їх синтезі необхідно передбачити такі вузли: регістри *R*1, *R*2, *R*3, які забезпечують тимчасове зберігання багаторозрядних двійкових чисел множеного, множника і часткових добутків, а також їх зсув на 1 розряд за 1 тактовий період; суматор *S* та віднімаючий лічильник *CT*, що фіксує кількість циклів.

В регістр $R1$ помістимо “n” – розрядне множене, в регістр $R2$ – “n” – розрядний множник, а “$n+1$” – розрядний регістр $R3$ будемо використовувати для накопичення суми часткових добутків. Перед додаванням до вмісту регістра $R3$ чергового часткового добутку попередня сума зсувається на один розряд вправо. Якщо частковий добуток дорівнює множеному, то суматор S додає вміст регістрів $R3$ і $R1$, а отримана сума пересилається в регістр $R3$.

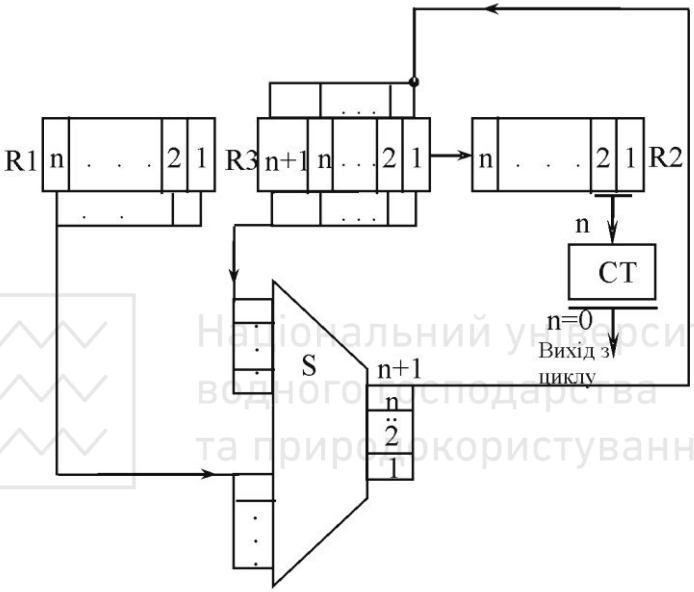


Рис. 37. Структурна схема операційного пристрою множення двійкових чисел

Одночасно із зсувом вмісту $R3$ зсувається вправо також вміст $R2$. При кожному такому зсуві в молодший розряд $R2$ надходить черговий розряд множника, по якому визначається черговий частковий добуток (рівний нулю або множеному).

Звільнений старший розряд регістра $R2$ заповнюється при зсуві молодшим розрядом регістра $R3$.

Виходячи зі словесного опису, будуємо структурну схему алгоритму операції множення (рис. 36).

В блоці 1 показаний вихідний стан регістрів $R1$, $R2$, $R3$, причому в регістрі $R3$, всі розряди знаходяться в нульовому стані, а в лічильник занесено число “n”$_{(2)}$.

При побудові структурної схеми алгоритму для позначення вмісту регістра його найменування поміщується в круглі дужки, а для вказання певного розряда в прямих дужках вказується його номер.

В блоці 2 перевіряється значення молодшого розряда числа в регістрі $R2$: ($R2[1]$); якщо в ньому записана “1”, то в блоці 3 до вмісту регістру R3 додається множене ($R1$), після чого в блоці 4 відбувається зсув вправо вмісту регістрів $R3$ і $R2$: $R3$, $R2 \leftarrow RRC$ ($R3$, $R2$). Якщо ($R2[1]$)=0, то частковий добуток дорівнює “0” і блок додавання 3 обходиться. Після n – кратного повтору цих дій в регістрі $R3$ створюється група старших розрядів добутку, а в регістрі $R2$ – група його молодших розрядів. Для підрахунку кількості циклів передбачений віднімаючий лічильник CT, в який попередньо (в блоці 1) записується двійкове число “n”$_{(2)}$. Далі, після кожного повтору цикла із вмісту лічильника віднімається “1” і перевіряється (в блоці 5), чи дорівнює нулю число у лічильнику. Якщо двійкове число у лічильнику ще не дорівнює нулю, то відбувається повтор дій в циклі, а при нульовому значенні числа у лічильнику відбувається припинення дій і вихід з циклу.

Виходячи з блок-схеми алгоритма, будуємо структурну схему операційного пристрою множення (рис.37).

Приклад. Розглянемо послідовність дій при множенні двійкових чисел $1101_{(2)} \times 1011_{(2)}=1000\ 1111_{(2)}$, згідно алгоритма (рис. 36).

Таблиця 22

Множене (R1)	Старші розряди добутку (R3)	Множник і молодші розряди добутку (R2)	Дія, що виконується
1 1 0 1	0 0 0 0	1 0 1 1	Вихідний стан
	+ 1 1 0 1		
	0 ← 1 1 0 1		Додавання
	0 0 1 1 0	→ 1 1 0 1	Зсув (R3) і (R2)
	+ 1 1 0 1		
	1 ← 0 0 1 1		Додавання
	0 1 0 0 1	→ 1 1 1 0	Зсув (R3) і (R2)
	0 0 1 0 0	→ 1 1 1 1	Зсув (R3) і (R2)
	+ 1 1 0 1		
	1 ← 0 0 0 1		Додавання
	0 1 0 0 0	→ 1 1 1 1	Зсув (R3) і (R2)
	1 0 0 0 1 1 1 1		Добуток

12.3.Побудова графа послідовного цифрового пристрою

Розглянемо побудову графа послідовного цифрового пристрою на прикладі асинхронного підсумовуючого лічильника.Його основним параметром є модуль рахунку, який визначає максимальну кількість одиничних імпульсів, що може бути порахована цим лічильником.Такі лічильники є послідовним сполученнням універсальних JK-тригерів, що працюють в лічильному режимі (рис.15).Сигнал для рахунку подається на перший тригер (нульового розряду).

Лічильник, що має "n" двійкових розрядів, може знаходитись в станах $0,1,2,3,...2^{n-1}$. При надходженні на його вхід 2^n-го імпульса лічильник повертається зі стану "2^{n-1}" в нульовий стан.

Побудуємо граф трирозрядного лічильника(рис.18), що здійснює рахунок імпульсів від "0" до "7".Для цього, попередньо, виходячи з його часової діаграми (рис.19), складаємо таблицю станів(табл.23).Заднім фронтом 1-го імпульсу перший тригер (нульового розряду) встановлюється в одиничний стан і на виходах лічильника з'являється двійкова комбінація $N_{(2)}=001$, що відповідає десятковому числу $N_{(10)}=1$. Після надходження другого імпульса його задній фронт скидає перший тригер у нульовий стан, а задній фронт вихідного імпульсу, що виникає на виході першого тригера, встановлює в одиничний стан другий тригер (першого розряду). При цьому на виходах лічильника створюється двійкова комбінація $N_{(2)}=010(N_{(10)}=2)$. Далі стани лічильника змінюються аналогічно до моменту надходження восьмого імпульса, заднім фронтом якого скидається в нульовий стан перший тригер, що спричиняє в свою чергу скид в нульові стани другого і третього тригерів. Після цього процес рахунку поновлюється.

Таблиця 23. Таблиця станів трирозрядного лічильника

Кількість вхідних імпульсів, n	Стани лічильника		
	U_2	U_1	U_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Для побудови графа стани пристрою позначаємо як вершини.Кількість вершин відповідає кількості станів лічильника і дорівнює 8. Дуги графа відзначаємо функціями переходу.Перехід відбувається при наявності сигналу перенесення(x=1), а при його відсутності($\overline{x}$ =0)-лічильник зберігає поточний стан. Отриманий кільцевий граф наведений на рис.38.

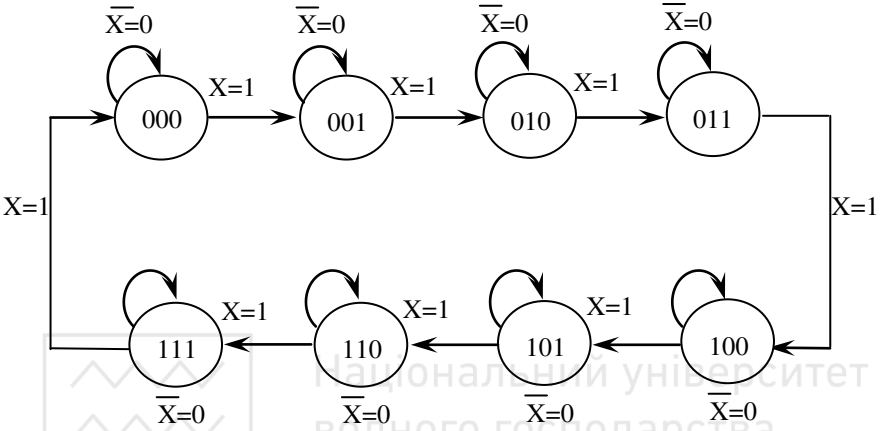


Рис.38.Кільцевий граф послідовного цифрового пристрою-асинхронного підсумовуючого лічильника

13. Архітектура та модульні пристрої мікропроцесорних систем

13.1. Архітектура базової мікропроцесорної системи (МПС)

Архітектура МПС – це логічна організація її апаратних засобів та програмного забезпечення для виконання заданих функцій. Принцип побудови базової МПС наведений на її структурній схемі (рис. 39).

Вона складається, як згадувалось вище, з пристрою введення (ПВ), мікропроцесора (МП), оперативного запам’ятовуючого пристрою (ОЗП), постійного перепрограмованого запам’ятовуючого пристрою (ППЗП) та пристрою виведення (ПВВ). Ці модульні пристрої з’єднуються між собою, за допомогою системи трьох шин. МП містить внутрішні регістри пам’яті швидкого доступу.

Адресна шина (ША) функціонує тільки в одному напрямі – від мікропроцесора до інших модулей системи і призначена для передавання адрес комірок пам’яті або номерів елементів

периферійних пристроїв. Шина даних (ШД) є двонапрямленою, тобто інформація по ній може передаватись у двох напрямках – від мікропроцесора до пристроїв пам'яті або до пристрою виведення і від пристроїв пам'яті або пристрою введення до мікропроцесора. Шина керування (ШК) призначена для передавання керуючих сигналів, що дозволяють записувати у пам"ять або зчитувати з пам'яті певні дані, вводити або виводити інформацію.Робота системи синхронізується генератором тактових імпульсів (ГТІ).

Розглянемо структурну схему МПС та її функціонування на прикладі виконання таких команд: IN (введення даних через ПВ), STORE (зберігання даних в ОЗП), OUT (виведення даних через ПВВ):

1. МП виставляє адресу "n" на ША і по ШК активізує вхід $\overline{R}$ ППЗП;
2. ППЗП виставляє код операції (КОП) IN на ШД, а МП приймає кодовану інформацію, розміщує її у внутрішньому регістрі та встановлює, що потрібний операнд;
3. МП виставляє адресу "$n + 1$" на ША і по ШК активізує вхід $\overline{R}$ ППЗП;
4. ППЗП виставляє операнд команди (з наступної комірки) на ШД, який приймається МП, і команда дешифрується повністю(операнд містить адресу порта введення ПВ);
5. МП за допомогою ША та ШК зчитує інформацію з порту введення ПВ і розміщує її у внутрішньому регістрі;
6. МП виставляє адресу "$n + 2$" на ША і по ШК активізує вхід $\overline{R}$ ППЗП;
7. КОП STORE зчитується МП з ШД і розміщується у його внутрішньому регістрі;
8. МП дешифрує КОП, виставляє адресу "$n + 3$" і по ШК активізує вхід $\overline{R}$ ППЗП;
9. Код операнда з адресою комірки ОЗП виставляється на ШД, зчитується МП, і дешифрується . Таким чином команда STORE повністю зчитана і декодована.
10. Процес виконання команди STORE починається з того, що МП виставляє на ША адресу комірки ОЗП і активізує вхід запису в ОЗП $\overline{W}$;
11. МП зчитує інформацію з внутрішнього регістру, виставляє її на ШД і записує в комірку ОЗП за виділеною адресою;
12. Після цього МП зчитує з ППЗП наступну команду OUT, виставляє адресу "$n + 4$" на ША і по ШК активізує вхід $\overline{R}$ ППЗП;
13. ППЗП виставляє код команди OUT на ШД, МП приймає її, дешифрує та встановлює, що потрібний операнд;
14. Лічильник МП встановлює на ША адресу "$n + 5$" і активізує вхід $\overline{R}$ ППЗП;

15. ППЗП виставляє код операнда OUT на ШД, МП приймає його і розміщує у внутрішньому регістрі;

16. МП повністю декодує команду, активізує виділений порт ПВВ за допомогою ША і ШК. Потім МП розміщує дані на ШД і пересилає у виділений порт ПВВ, звідки вони надходять на відеотермінал, або виконавчі пристрої.

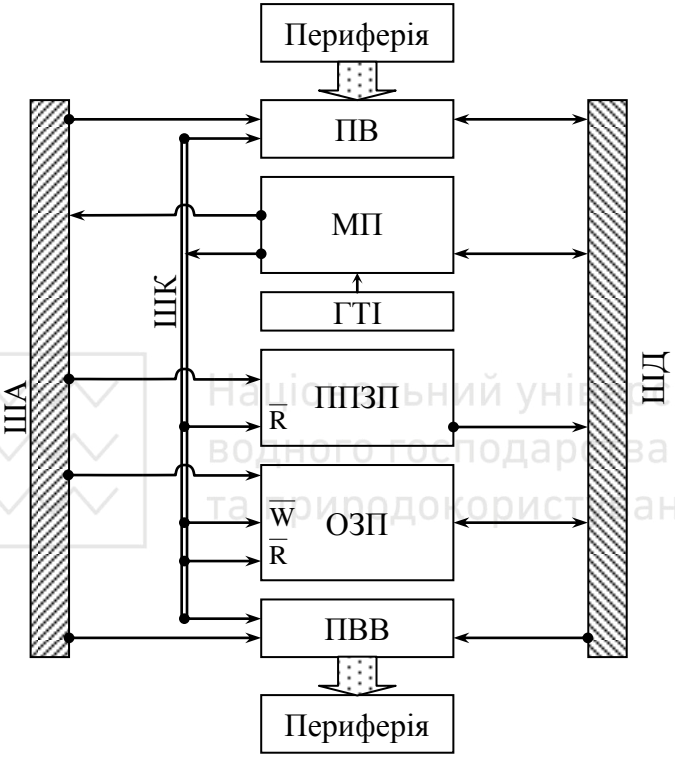


Рис.39. Структурна схема базової мікропроцесорної системи

Таким чином, пристрій введення (ПВ) забезпечує розміщення інформації, що надходить у мікроконтролер, та керування її введенням. Мікропроцесор (МП) керує функціонуванням мікроконтролера, проводить обробку інформації згідно керуючої програми і в залежності від отриманих результатів формує набір керуючих сигналів. Оперативний запам'ятовуючий пристрій (ОЗП) використовується для тимчасового розміщення даних і забезпечує як запис, так і зчитування інформації. Постійний перепрограмований запам'ятовуючий пристрій (ППЗП) призначений для зберігання програми – монітора, керуючої програми, а також основних

констант технологічного процесу і розрахований тільки для зчитування даних. Пристрій виведення (ПВВ) призначений для розміщення даних, отриманих внаслідок обробки інформації та формування керуючих сигналів. В залежності від результатів обробки інформації мікропроцесор формує набір керуючих сигналів, які надходять в оперативний запам'ятовуючий пристрій, а потім пересилаються і розміщуються в пристрої виведення.

13.2. Базовий мікропроцесор, його структура та функціонування

Розглянемо будову і функціонування мікропроцесора, який є основним елементом МПС, на прикладі базового мікропроцесора КР 580 (зарубіжний аналог Intel 8080). Мікропроцесор (МП) – це центральний процесор (СР), реалізований на одній мікросхемі великого степеня інтеграції (ВІС). Структурна схема мікропроцесора (рис.40) дає можливість наочно розглянути його будову і роботу по виконанню основних функцій. Він містить арифметико – логічний пристрій, регістри, пристрій керування і синхронізації та допоміжні елементи, які з'єднані внутрішньою шиною даних (ВШД).

Арифметико – логічний пристрій (АЛП) виконує арифметичні та логічні операції над даними, що надходять на його входи. До арифметичних належать операції додавання, віднімання і зсуву восьмирозрядних двійкових чисел. Для виконання множення, ділення, піднесення до степеня тощо вже треба складати підпрограми. До логічних належать операції І, АБО, НІ та їх комбінації. Складніші операції, наприклад порівняння двох чисел виконують за допомогою підпрограми. АЛП працює разом з такими регістрами: акумулятором, буферними регістрами тимчасового зберігання та регістром стану.

Він має інформаційні та керуючі входи, на які надходять дані і коди операцій, та забезпечує триступеневе перетворення інформації. На першому ступені виконуються логічні операції і формуються напівсуми. На другому – відбувається генерація перенесень і на третьому ступені формується повна сума (рис. 23,24; табл. 20-21).

На виході першого ступеня формується функція напівсуми, а на виході другого ступеня - функція генерації перенесення в даному розряді і функція передачі перенесення. Функція напівсуми призначена для формування результату додавання з врахуванням перенесення із сусіднього молодшого розряду.

Один з керуючих входів задає тип операції і при низькому рівні напруги на цьому вході (логічний "0"),- виконуються логічні операції, а при її високому рівні (логічна „1") – арифметичні операції. При виконанні логічних операцій, перенесення не відбувається.

На третьому ступені із напівсум, що формуються на першому ступені, та перенесень, що генеруються на другому ступені, формується повна сума.

Акумулятор (А) – спеціальний однобайтовий регістр, який є основною операційною ланкою АЛП під час маніпуляцій з даними. Більшість арифметичних і логічних операцій передбачає попереднє розміщення одного

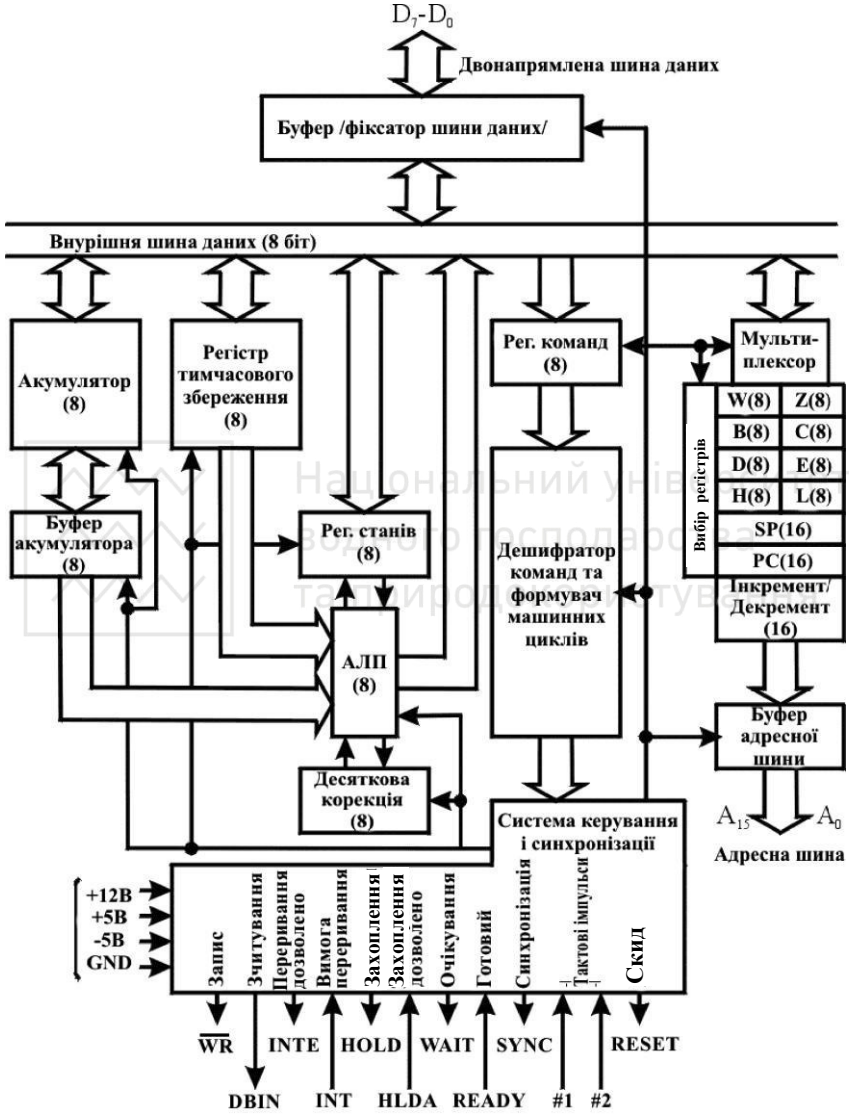


Рис. 40. Структурна схема базового мікропроцесора

з операндів в А, а другого в буферному регістрі тимчасового зберігання. Результат виконаної операції також розміщується в А, після чого може бути виведений на зовнішній пристрій або збережений в ОЗП (рис. 41).

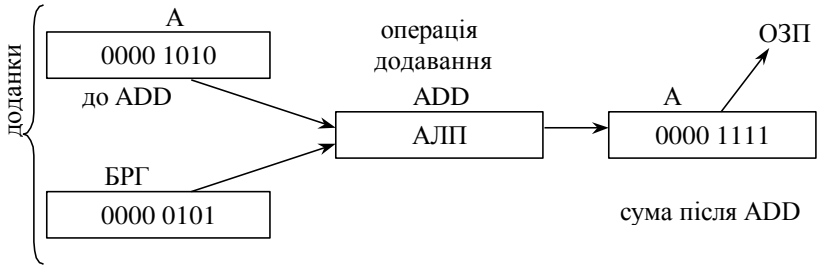


Рис. 41. Операція додавання з використанням акумулятора (А)

Буферні регістри (БРГ) призначені для тимчасового розміщення даних перед їх введенням в АЛП, поки пристрій не завершить обробку даних, які надійшли раніше.

Відповідні розряди регістра стану – флаги визначають характеристики розміщеного в акумуляторі результата виконаної операції. Регістр складається з п'яти флагів (тригерів):

1. "С" або „CY" (carry) – флаг перенесення, встановлюється в стан "1", коли результат обчислення виходить за межі 8 розрядів;
2. "Z" (zero) – флаг нуля, встановлюється в стан "1" при наявності нульового результату;
3. "AC" (auxilary carry) – флаг допоміжного перенесення, встановлюється в стан "1", якщо двійкове число, що було в акумуляторі, переносилось з третього в четвертий розряд;
4. "P" (parity) – флаг парності, встановлюється в стан "1" при парній кількості одиниць у числі, записаному в акумуляторі.
5. "S" (sign) – флаг знака встановлюється в стан "1", якщо в старшому розряді байта даних з'являється "1", що вказує на від'ємний результат.

Регістри загального призначення B, C, D, E, H, L є однобайтовими. Вони доступні для програмування і використовуються для тимчасового зберігання даних та проміжних результатів обчислень, що виконуються АЛП, а також адрес. Їх можна використовувати також, як три пари 16 – розрядних регістрів BC, DE, HL (наприклад для зберігання адрес комірок пам'яті).

Регістри W, Z є програмно недоступними і забезпечують тимчасове зберігання даних при виконанні команд.

Вказівник стеку (SP) – двобайтовий 16-розрядний регістр . В ньому зберігається адреса поточної комірки стеку. Стек – це ділянка оперативної

пам'яті для тимчасового зберігання адрес команд та вмісту регістрів при переходах до підпрограм та виходу з них (операціях переривання). Дані від МП надходять у верхню частину стекової пам'яті і при цьому вказівник стеку зменшується на "1", щоб завжди вказувати на адресу останньої заповненої комірки стека. При зчитуванні даних зі стеку вміст вказівника збільшується на "1" з кожним вибраним байтом.

Лічильник команд або програмний лічильник (РС) – це двобайтовий 16 – розрядний регістр, в якому формується адреса команди, що повинна виконуватись. Він автоматично забезпечує приріст адреси і завжди вказує на 1-й байт наступної команди.

Система керування і синхронізації (СКС) – виробляє керуючі сигнали для всіх вузлів мікропроцесора при виконанні команд. Робота схеми керування мікропрограмується. Виконання кожної команди відбувається у послідовності, визначеної кодом операції, і синхронізується в часі сигналами Ф1 та Ф2 тактового генератора. Період синхроімпульсів є машинним тактом (тривалість 0,5÷2мкс). Машинний цикл – це час, необхідний для вилучення одного байта інформації з пам'яті або виконання команди, визначаємої одним машинним словом. Він дорівнює кільком машинним тактам. Час виконання однієї команди визначається часом отримання, дешифрування та виконання команди і складає кілька машинних циклів. СКС працює разом з регістром команд та дешифратором команд.

Регістр команд (RC) – це однобайтовий регістр, призначений для тимчасового зберігання кода операції (КОП), поки мікропроцесор не виконає попередню команду. Після виконання поточної команди в RC автоматично записується КОП із пам'яті, адреса якого знаходиться у лічильнику команд РС.

Дешифратор команд (DCC) дешифрує КОП та визначає мікропрограму для виконання необхідної команди системою керування і синхронізації (СКС) після чого вводить її в дію.

Базовий мікропроцесор забезпечує:

– пересилання байта даних від пристрою введення, або до пристрою виведення;

– зчитування байта даних з пам'яті та запис у пам'ять;

– виконання операцій згідно керуючої програми, що записана в пам'яті.

При цьому кожна операція реалізується мікропроцесором як певна послідовність мікроопераций. 8-розрядний двійковий код операції, що зчитується з пам'яті, надходить в регістр команд RC, де зберігається на протязі її виконання. По результату його дешифрування СКС формує послідовність мікроопераций, які створюють певну операцію.

Команда складається з операції та операнду (дані, адреса) і виконується від 1 до 5 машинних циклів. Базовий мікропроцесор має такі машинні цикли: вибір команди, читання з пам'яті, запис у пам'ять, читання зі

стеку, запис у стек, введення із зовнішнього пристрою, виведення на зовнішній пристрій, переривання, зупинка, переривання під час зупинки. При цьому кожний машинний цикл „*C*" містить 3-5 тактів (періодів) „*T*".

Розглянемо команду LDA (завантажити дані в акумулятор), яка переписує в акумулятор дані з комірки пам'яті:

№	Цикли												
	*C*1				*C*2			*C*3			*C*4		
Такти	*T*1	*T*2	*T*3	*T*4	*T*1	*T*2	*T*3	*T*1	*T*2	*T*3	*T*1	*T*2	*T*3

Під час циклів (*C*) і тактів (*T*) мікропроцесор виконує такі мікропрограми і мікрооперації:

*C*1 – вибірку коду операції (код операції розміщується в *RC*):

T1 – вміст *PC* видається через буфер шини адреси БША в ША;

*T*2 – зчитування вмісту комірки пам'яті;

*T*3 – дані з пам'яті по ШД пересилаються в *RC*;

*T*4 – КОП передається в *DCC* для формування послідовності мікрооперацій, вміст *PC* збільшується на 1;

*C*2 – звернення до пам'яті, 8 молодших розрядів адреси вибираються та розміщуються в регістрі *Z*:

*T*1 – вміст *PC* через БША виставляється на ША;

*T*2 – зчитування вмісту комірки пам'яті;

*T*3 – вміст комірки пам'яті по ШД передається в регістр *Z,* а вміст *PC* збільшується на 1;

*C*3 – повторне звернення до пам'яті, 8 старших розрядів адреси вибираються та пересилаються в регістр *W*:

*T*1 – новий вміст РС через БША виставляється на ША;

*T*2 – зчитування вмісту комірки пам'яті;

*T*3 – вміст комірки пам'яті по ШД передається в регістр *W,* а вміст *PC* збільшується на 1;

*C*4 – завантаження *A* даними з пам'яті згідно адресі, що розміщується в регістрах *W*, *Z*:

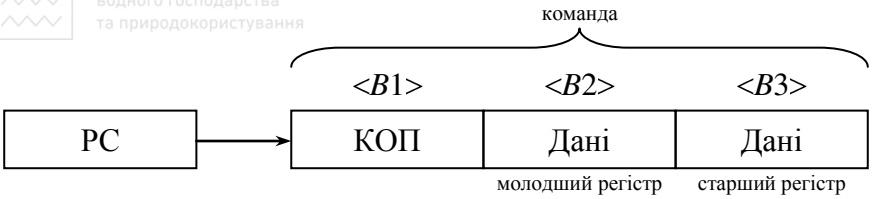
*T*1 – вміст *W* і *Z* передається в ША;

*T*2 – зчитування з пам'яті вмісту комірок за адресою *W* і *Z*;

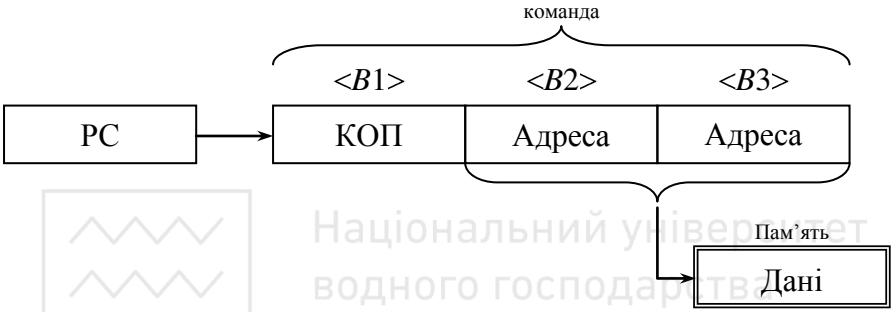
*T*3 – вміст комірок пам'яті по ШД пересилається в *A*.

В базовому мікропроцесорі використовуються такі способи адресації:

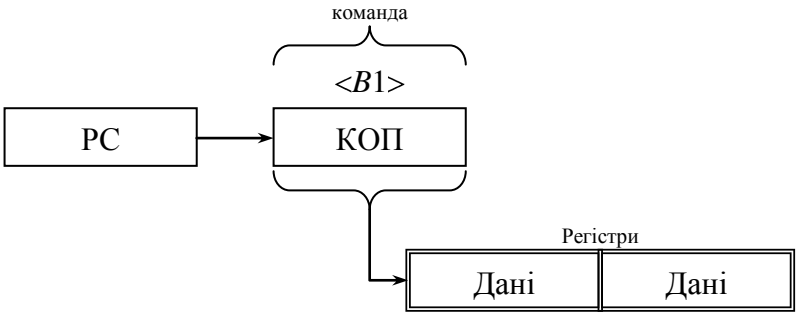
1. **Безпосередня**, яка є найбільш економічним способом зберігання та пошуку інформації, тому що команда містить безпосередньо дані:



2. **Пряма**, яка є менш економічною, але дає можливість адресуватись до будь-якої комірки адресного простору пам'яті:

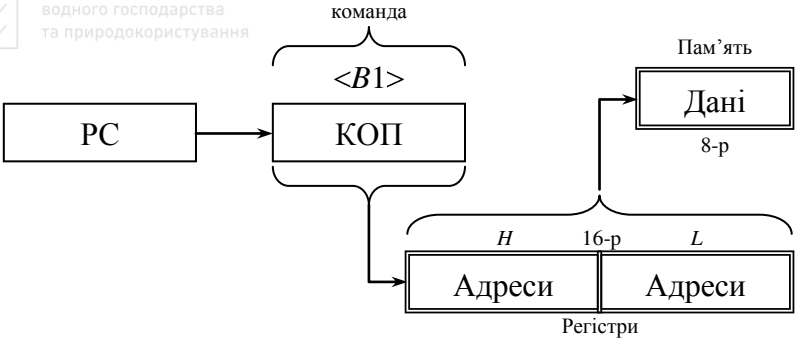


3. **Регістрова**, при якій КОП вказує на регістри, що містять дані. Можливість визначення пари регістрів в 1-байтовій команді реалізується за



рахунок того, що адреси регістрів є 3-розрядними двійковими числами.

4. **Посередня адресація**, при якій в регістровій парі, що визначається КОП, знаходиться повна 16-розрядна адреса комірки пам'яті, яка містить дані. Старший байт записується в першому регістрі пари, а молодший – в другому. Вказівниками адреси при цьому є пари регістрів:



При пересиланнях даних з регістрів та пам'яті розрізняють джерела *S* (source) та приймачі *D* (destination). Коди регістрів та регістру станів є фіксованими. Умови в мнемокоді команди в залежності від значень флагів дешифруються так:

Z – результат нульовий;
NZ – результат ненульовий;
NC – немає переносу;
C – є перенос;
PO – немає парності;
PE – є парність;
P – результат позитивний;
M – результат від'ємний.

Мікропроцесори є функціонально-обмеженими і тому працюють в системах разом з модульними пристроями пам'яті, які діляться на оперативні (ОЗП) та постійні запам'ятовуючі пристрої (ПЗП).

13.3. Статичні оперативні запам'ятовуючі пристрої

Оперативні запам'ятовуючі пристрої **ОЗП** (RAM – random access memory) призначені для тимчасового розміщення інформації і використовуються як для запису (*W*), так і для зчитування ($\overline{R}$) даних з пам'яті. Існують два типи ОЗП: статичні і динамічні. Статичні ОЗП побудовані на комірках пам'яті, організованих в певний масив. Кожна комірка містить тригер на комплементарних МДН-транзисторних парах з вузлом керування (рис. 42). Їх вибір здійснюється за двомірною адресацією.

Щоб вибрати комірку з пам'яті, треба одночасно подати рівень логічної "1" на адресні входи *A*1, *A*2, з яких він надходить на схеми *D*2, *D*3 і *D*5. Під час запису рівень логічної "1" подається на вхід "запис-читання"

$W/\overline{R}$ і сигнал з входу даних D через схему $D2$ (якщо на вході логічна “1”) або через схеми $D1$, $D3$ (якщо на вході логічний “0”) записується в тригер $D4$.

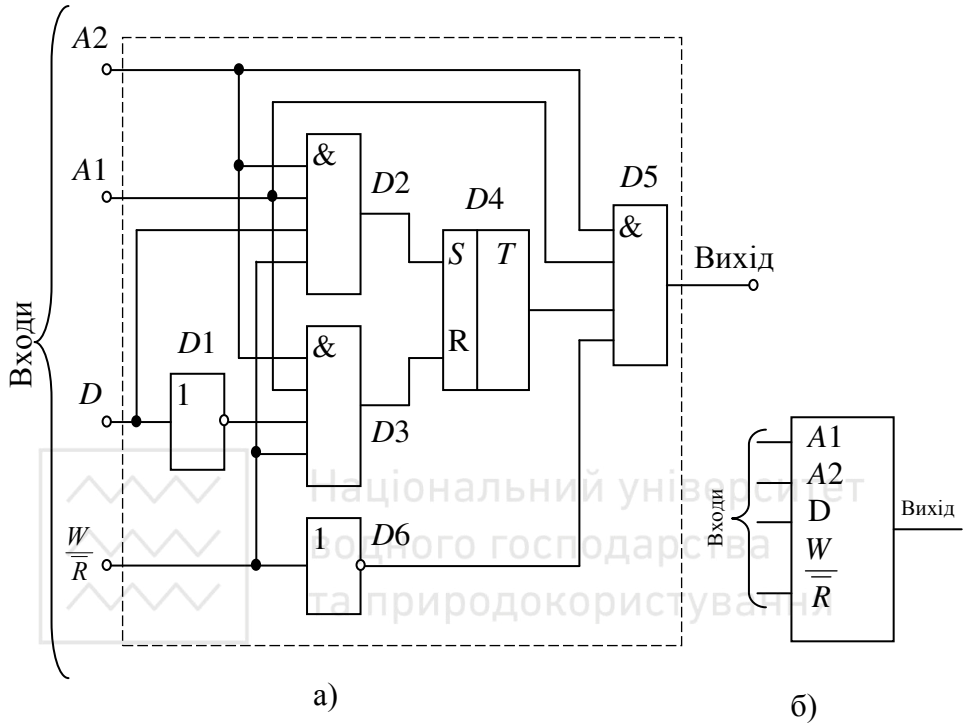


Рис. 42. Комірка пам’яті (КП): а) структурна схема; б) умовне позначення

При цьому вихідна схема $D5$ закрита логічним “0”, що надходить з виходу $D6$. Під час читання на вхід $W/\overline{R}$ подається логічний “0” і вихідний сигнал з тригера $D4$ передається на вихід $D5$.

Модулі ОЗП складаються з комірок пам’яті КП (рис. 42), які є базовими елементами і з’єднуються за схемою з двомірною адресацією у вигляді квадратної матриці (рис. 43). Модуль містить також керуючі кола, які забезпечують роботу КП у режимах запису інформації W, її зберігання $\overline{CS}$ та читання $\overline{R}$. Конструктивно модулі виконуються у вигляді окремих мікросхем з організацією $N\times1$ (N – кількість однорозрядних комірок). Наприклад, якщо матриця складається з 16 рядків та 16 стовпців, то ємність її пам’яті дорівнює $16\times16=256$ біт, а її організація 256×1. Регістр адрес містить дві групи розрядів: n_1 розрядів визначає двійковий номер рядка, а n_2 –

двійковий номер стовпця, в якому розташована обрана комірка пам'яті. Використовуючи відповідну кількість мікросхем та з'єднуючи їх між собою можна побудувати оперативну пам'ять з необхідною організацією. При побудові ОЗП ємність його пам'яті змінюють за рахунок нарощування розрядності, тобто збільшення кількості мікросхем, що створюють окреме слово, та нарощування кількості слів у пам'яті. Для нарощування розрядності, на мікросхеми, які створюють так званий ряд (або лінійку), подають одну і ту ж адресу та ідентичні сигнали дозволу записування інформації W /$\overline{\text{R}}$ і вибору мікросхеми $\overline{CS}$. Завдяки цьому сукупність кількох мікросхем ряду функціонує як одна мікросхема (рис. 44).

При зчитуванні інформації на виході кожної з мікросхем з'являється певний розряд слова, а при її записуванні вхідне слово порозрядно заноситься в комірки пам'яті окремих мікросхем. Таким чином, якщо організація мікросхеми $N\times1$ (тобто N однорозрядних комірок), то для слова пам'яті з організацією $N\times n$ потрібно n мікросхем. Наприклад, якщо необхідно побудувати на мікросхемах 1024×1 ($K\times1$) ряд, що реалізує 8-розрядне слово, то кількість мікросхем $n=8$. Для побудови ОЗП нарощують кількість слів за рахунок збільшення кількості рядів та загальної кількості мікросхем (рис. 45). Розряди адрес ОЗП діляться на дві групи $A1$ та $A2$. Група розрядів $A2$ визначає номер ряду, а група розрядів $A1$ – номер комірки у вибраному ряді. Вибір ряду здійснюється за допомогою дешифратора DC, на вхід якого подається $A2$, а кожний з його виходів під'єднується до входів $\overline{CS}$ відповідного ряду. В залежності від кодової комбінації, яку містить $A2$, на відповідному виході дешифратора з'являється рівень логічного "0", який забезпечує вибір необхідного ряду мікросхем.

При зростанні ємності пам'яті МПС,-знижується його швидкодія внаслідок зростання часу пошуку інформації в пам'яті. Для підвищення швидкості обміну інформацією між мікропроцесором та пам'ятю, передбачається буферна *кеш-пам'ять* з невеликою ємністю, яка будується на основі оперативної пам'яті статичного типу. Її робота грунтується на принципах часової та просторової локальності, суттю яких є велика ймовірність звернення програми на протязі невеликого проміжку часу до групи суміжних комірок. Згідно з принципом часової локальності, інформацію у кеш-пам'яті слід зберігати протягом певного часу, а принцип просторової локальності вказує на доцільність розміщення у кеш-пам'яті певного блоку інформації, незначного за обсягом (3÷5 команд). При зчитуванні даних з пам'яті, мікропроцесор звертається спочатку до кеш-пам'яті та перевіряє, чи містить вона необхідну інформацію. Цей процес порівнює старші розряди адрес, які мікропроцесор виставляє на адресну шину. Співпадання цих величин вказує на те, що в кеш-пам'яті зберігаються необхідні дані. У цьому разі дані зчитуються з кеш-пам'яті. Якщо величини

не збігаються, то відповідні дані копіюються з оперативної пам'яті у кеш-пам'ять.

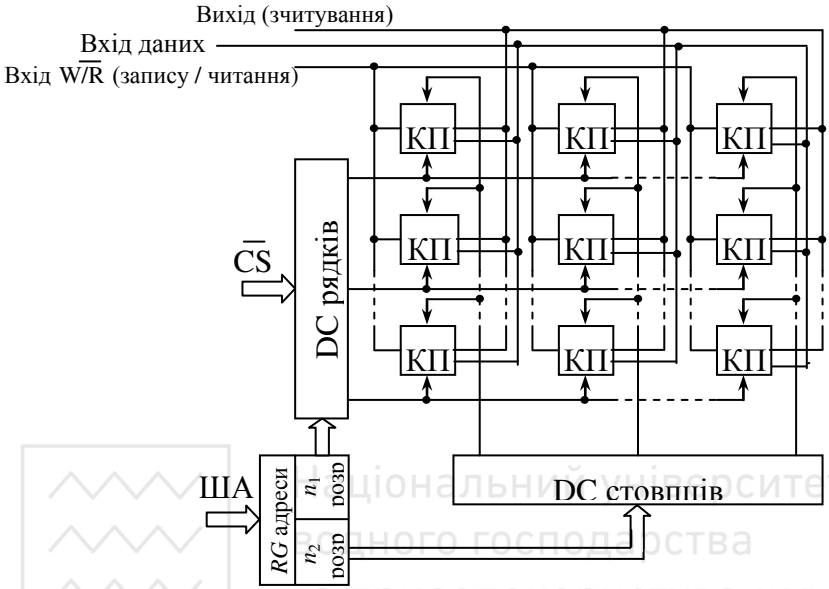


Рис. 43. Типова структура мікросхеми ОЗП: КП – комірка пам'яті; *RG* – регістр; *DC* – дешифратори; $\overline{CS}$ – вхід вибора мікросхеми.

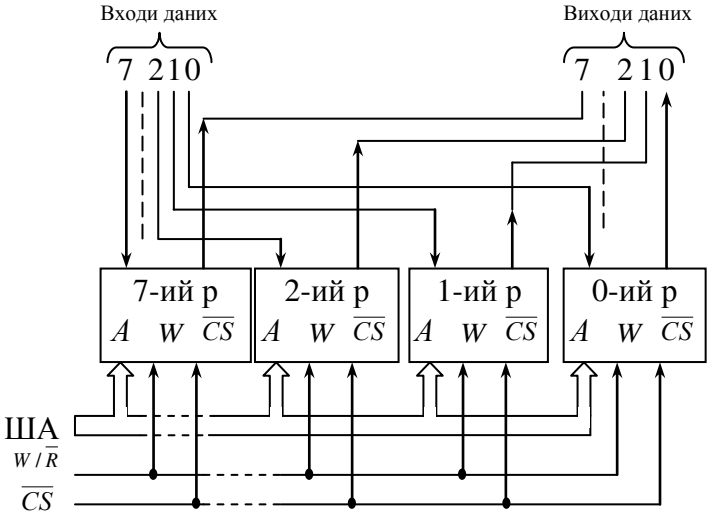


Рис.44. Схема лінійки (ряду мікросхем)

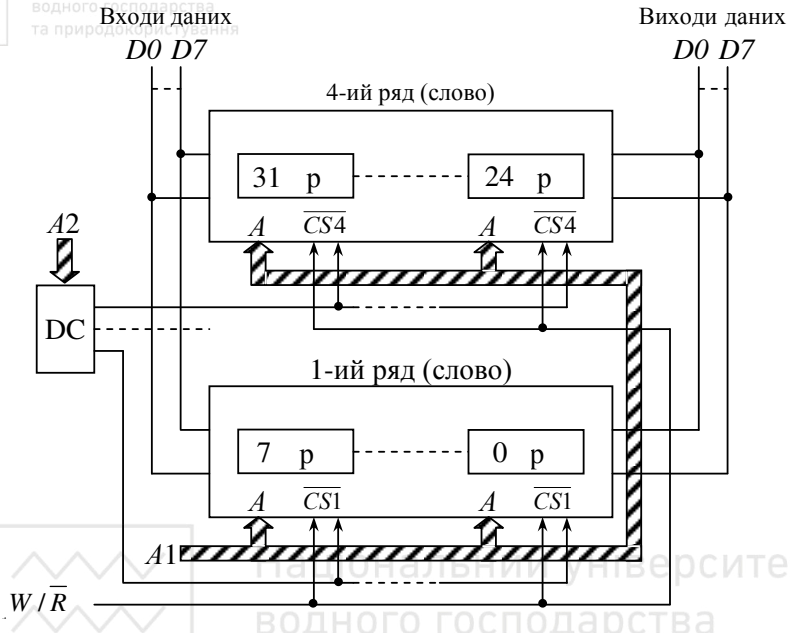


Рис.45. Структурна схема статичного ОЗП

13.4. Динамічні оперативні запам'ятовуючі пристрої

В динамічних ОЗП елементом пам'яті є вхідна ємність "С" МДН-транзистора. Якщо подати на вхід комірки динамічної пам'яті (рис.46) позитивний потенціал, а потім його зняти, то вхідна ємність "С"транзистора VT2 залишиться зарядженою, що відповідає стану логічної "1", а відсутність заряду на ній- відповідає стану логічного "0".

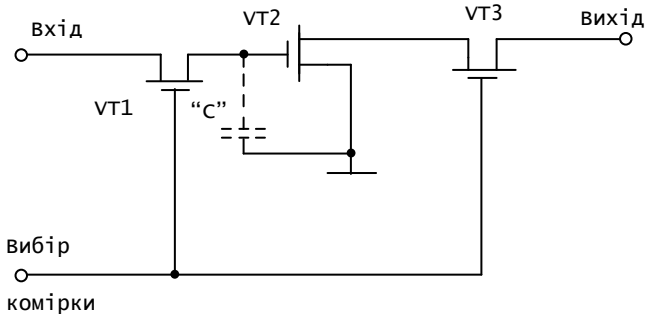


Рис. 46. Комірка динамічної пам'яті ОЗП на МДН-транзисторах VT1,VT2 та VT3

Але з часом заряд ємності "С" зменшується і тому потрібна регенерація- періодичний перезапис даних у комірку пам'яті.

Типова структура однобітної організації динамічної пам'яті N×1 з мультиплексною передачею адрес рядків і стовпців наведена на рис.47 . Розглянемо її функціонування на прикладі запам'ятовуючого пристрою ємністю N=64K×1. Для роботи з пам'яттю використовуються 8-розрядні регістри RG рядків і стовпців, виходи яких під'єднуються до входів відповідних дешифраторів DC. Адресація комірки виконується за два послідовних цикли . Спочатку на 8 адресних входів регістрів надходять сигнали молодшого байту адреси (A0÷A7), які супроводжує стробуючий сигнал $\overline{RAS}$(Row Adress Strobe-строб адреси рядка). При цьому 8 молодших розрядів адреси записуються в регістр дешифратора рядків. Потім на 8 адресних входів регістрів надходять сигнали старшого байту (A8÷A15) і стробуючий сигнал $\overline{CAS}$(Column Adress Strobe-строб адреси стовпця), внаслідок чого 8 старших розрядів адреси заносяться в регістр дешифратора стовпця. Тепер в обох регістрах записані 16 біт адреси, що надає можливість вибрати одну з 2^8*2^8=65536=64К комірок пам'яті. Так здійснюється мультиплексування адресних сигналів в часі. Модулі динамічних ОЗП організуються у вигляді двох банків суміжних байтів з парними та непарними адресами (рис. 77). Кожний з них складається з кількох динамічних пристроїв однобітової організації N*1 (N=64 К; 256 К; 1 М) (рис. 47).

Вони працюють в режимах запису, зчитування, зчитування/модифікації/запису, сторінкового запису, сторінкового зчитування та регенерації. Для керування динамічними ОЗП використовують контролери динамічної пам'яті, які формують адресні та керуючі сигнали в режимах роботи і регенерації.

Запис інформації відбувається заднім фронтом сигналу $\overline{CAS}$ при $W/\overline{R}=0$, а її зчитування–заднім фронтом сигналу $\overline{CAS}$ при $W/\overline{R}=1$. Режим, зчитування/модифікації/запису полягає в зчитуванні інформації з подальшим її записом в ті ж самі елементи пам'яті. Сторінкові режими запису та зчитування реалізуються зверненням до певного пристрою за адресою рядка, з наступним вибором елемента пам'яті цього рядка зміною адрес стовпців.

Регенерація інформації відбувається зверненням до кожного з рядків, при цьому формується адреса рядка і сигнал $\overline{RAS}$, а $\overline{CAS}$ дорівнює логічній "1".

13.5. Пам'ять програм. Постійні (ПЗП) та перепрограмовані постійні запам'ятовуючі пристрої (ППЗП)

На відміну від ОЗП ці пристрої є енергонезалежними, інформація в їх комірки записується однократно і після цього використовується тільки режим читання. За способом занесення інформації вони діляться на ПЗП, які програмовані маскою на підприємстві-виробнику ; ПЗП, які програмуються безпосередньо користувачем, а також ППЗП. Оптимальним з точки зору користування є ППЗП.

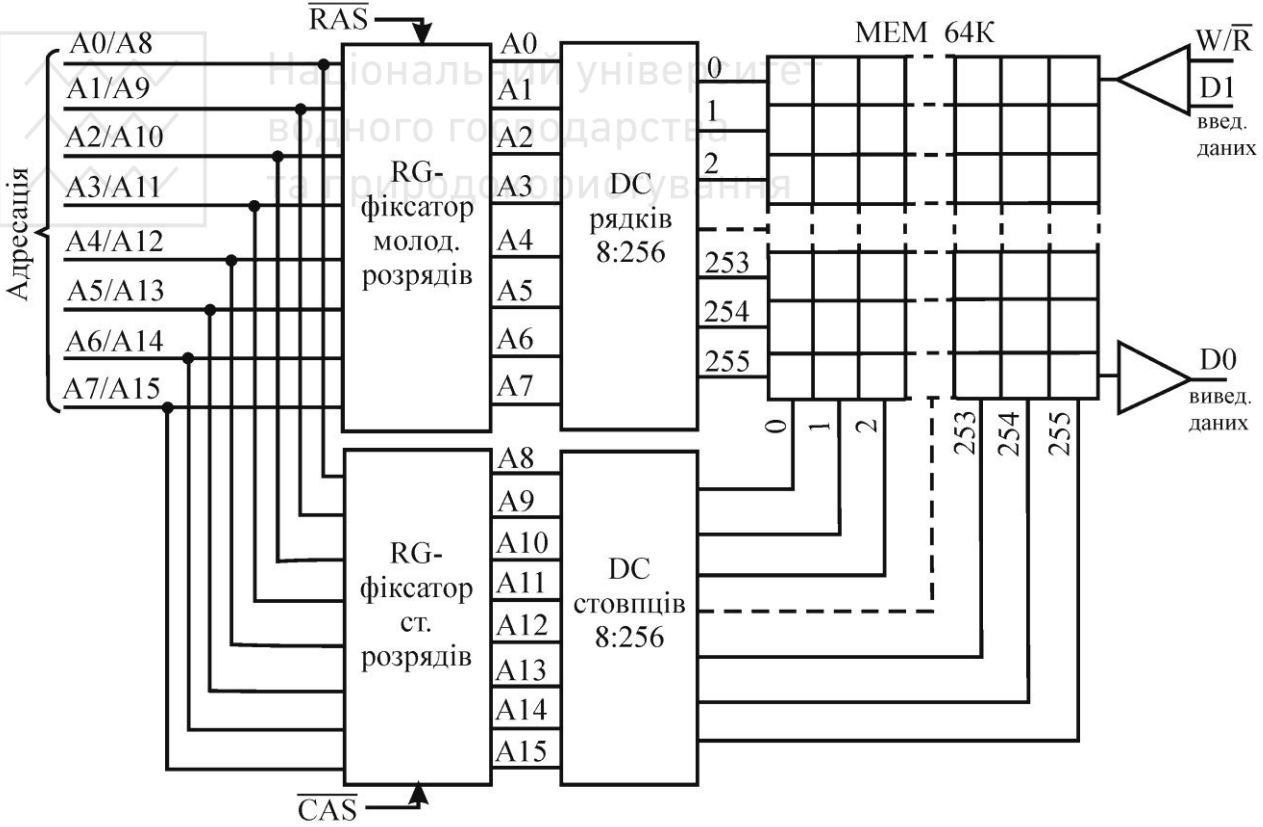


Рис. 47 Структурна схема динамічного ОЗП 64К*1

Розглянемо комірку пам'яті ППЗП з електричним записуванням інформації та її стиранням за допомогою ультрафіолетового опромінювання EP ROM(рис.49), яка містить МДН-транзистори VT1 і VT2.

Транзистор VT1 з ізольованим затвором використовується при виборі потрібної комірки пам'яті, а носієм інформації є транзистор VT2 з плаваючим затвором. Польовий МДН-транзистор VT2 (метал-діелектрик-напівпровідник) (рис.48) містить напівпровідникову підкладку n-типу - 1, в якій сформовані напівпровідникові зони р-типу , що створюють відповідно джерело - 2 (”Д”) та стік - 3 (”С”).Над проміжком між ними розміщений шар діелектрику (SiO_2) – 4, в якому сформований додатковий кремнієвий плаваючий затвор 5.

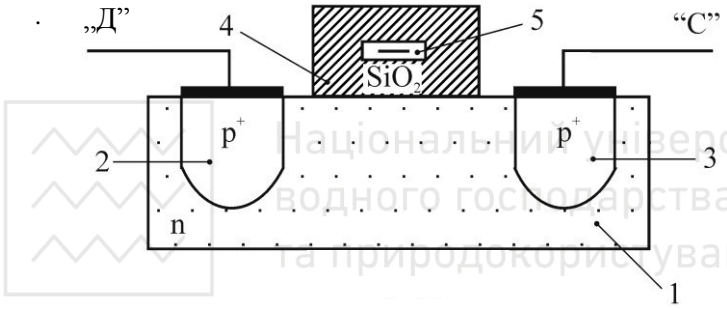


Рис.48. Структура МДН -транзистора VT2 з плаваючим затвором.

У вихідному стані перехід “джерело-стік” струм не проводить і транзистор VT2 знаходиться у стані логічної “1”. Для запису в нього логічного „0” на вивід „Д” або „С” подають імпульс підвищеної напруги (+25В). При цьому у зворотно зміщеному „p-n” – переході відбувається лавинне зростання кількості електронів та зарядка затвора “5” за рахунок їх часткової інжекції. Негативний заряд на затворі зберігається практично необмежений час. Він притягує дірки і індукує провідний “*p*”-канал між джерелом і стоком, внаслідок чого транзистор VT2 переходить у стан логічного “0”. Стирання інформації в мікросхемах ППЗП відбувається за рахунок ультрафіолетового опромінювання, через спеціальне кварцове скло, що приводить до стікання зарядів з плаваючих затворів транзисторів та їх переходу в непровідний стан. При зчитуванні сигнал з адресної шини відкриває VT1 і в залежності від стану VT2 на інформаційну шину надходить нульовий потенціал („0”) або залишається потенціал напруги живлення („1”) (рис. 49).

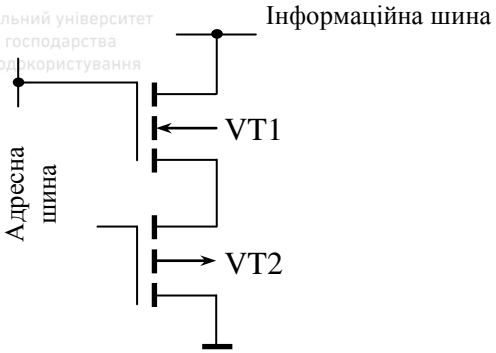


Рис. 49. Електрична схема комірки ППЗП

Більш досконалою є Flash-пам'ять (флеш-пам'ять) з електричним стиранням та перепрограмуванням. Її елементами є МДН-транзистори з плаваючими затворами, що виконані за технологією ETOX, розробленою фірмою Intel. Вони містять підкладку p-типу, на якій сформовано n-зони джерела і стоку. Над проміжком між джерелом і стоком розташований керуючий затвор, відділений від підкладки шаром діелектрику SiO_2. В ньому сформована область з полікремнію, що виконує функції плаваючого затвору, в якому накопичуються електрони і створюється негативний заряд. Якщо подати на керуючий затвор номінальну напругу і при цьому у плаваючому затворі відсутній негативний заряд, то МДН-транзистор є відкритим, а при накопиченні заряду у плаваючому затворі- він закривається.

Завдяки використанню технології фірми Intel в МДН-транзисторі утричі (до 100 A^0) зменшена товщина шару діелектрика між плаваючим затвором та підкладкою. Це надає можливість записувати інформацію при напрузі, зниженій до 12В. Разом з цим за рахунок дуже малої товщини шару p-n-переходу виникає тунельний ефект, тобто носії зарядів переходять з однієї області p-n-переходу в іншу без зміни своєї енергії , не долаючи потенціальний бар'єр.

Це надає можливість електричного стирання, тобто видалення заряду з плаваючого затвору за рахунок тунельного ефекту при напрузі між стоком і керуючим затвором $U_{C\text{-}З}$=12В. Кількість циклів стирання / запису Flash-пам'яті не менше 10^5 разів, напруга живлення U=5В і напруга програмування $U_{пр}$=12В.

Структура ПЗП (рис.50) будується як і структура ОЗП за матричною схемою. Як і ОЗП, матриця складається з комірок пам'яті КП, що утворюють рядки та стовпці. Але на відміну від ОЗП при зчитуванні видається вміст цілого рядка елементів пам'яті. Такий рядок містить кілька слів. За допомогою селектора із рядка виділяється і передається на вихід необхідне слово.

Наприклад, якщо ПЗП має об'єм пам'яті $M=2^{10}$ біт, то вона ділиться на $N=2^7$ слів по 2^3 розрядів (біт) у кожному слові. Матриця пам'яті містить $K=2^{10}=1024$ комірки пам'яті, розташованих в 32 рядках та 32 стовпцях. Адреса має 7 розрядів і ділиться на дві групи : п'ятирозрядну групу А1 та дворозрядну групу А2. Група А1 подається на дешифратор DC1, який вибирає один з $2^5=32$ рядків матриці пам'яті. Вміст рядка складається з 32 біт або 4-х восьмирозрядних слів. Номер слова в рядку задається групою А2. Дешифратор DC2 перетворює цю адресну групу в сигнал на одному з 4-х своїх виходів. По цьому сигналу в селекторі з вмісту рядка виділяється відповідне слово, яке через буфер введення-виведення передається на вихід мікросхеми.

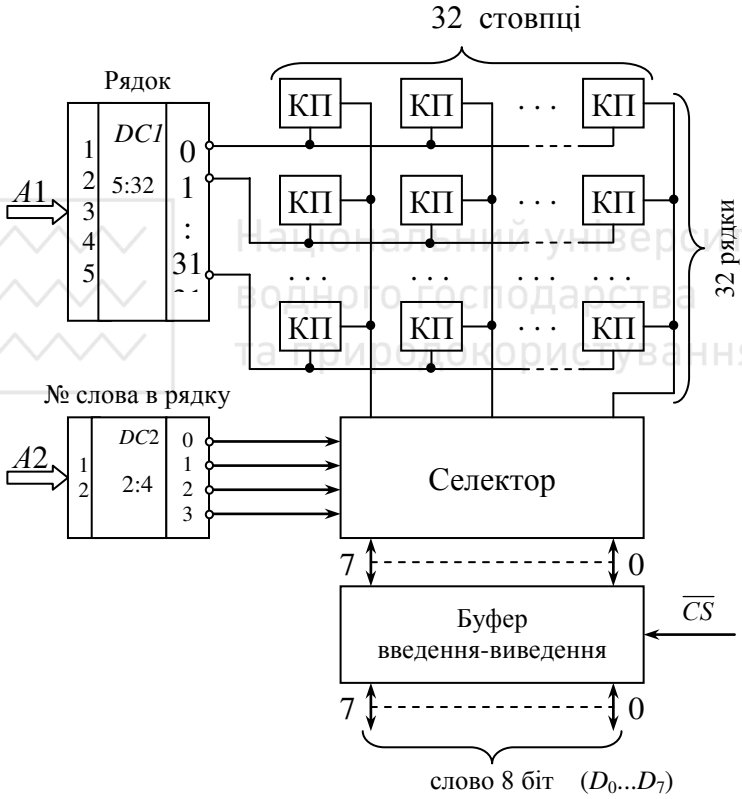


Рис.50. Структурна схема ПЗП

Виводи типової мікросхеми ПЗП наведені на рис. 51. В мікросхемах ППЗП додатково передбачаються виводи програмування $\overline{PR}$.

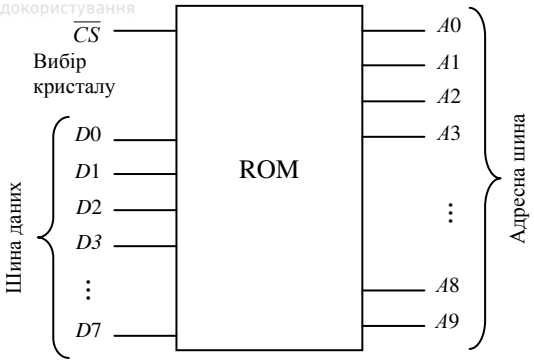


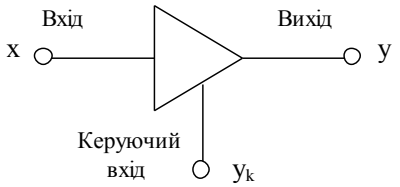
Рис. 51. Типова мікросхема ПЗП: $D0 \ldots D7$ – контакти виведення даних; $A0 \ldots A9$ – контакти введення адрес

13.6. Під'єднання до шин модульних пристроїв мікропроцесорних систем

Як наведено вище, шини мікропроцесорної системи (*BUS*) – це сукупність ліній передачі інформації, об'єднаних загальними функціональними ознаками: шина даних, шина адрес та шина керування. Шина даних (*date bus*) – це система ліній передачі даних, адресна шина (*adress bus*) – система передачі адрес та шина керування (*control bus*) – це система ліній передачі керуючих сигналів.

Під'єднання основних модулів мікропроцесорної системи: мікропроцесора, постійного та оперативного запам'ятовуючих пристроїв, а також пристроїв введення / виведення до шин здійснюється за допомогою двонапрямлених шинних формувачів (*bus driver*). Їх основними елементами є тристабільні елементи (драйвери), які є шинними підсилювачами. Умовне позначення таких підсилювачів та їх таблиці істинності наведені на рис. 52 а, б; 53 а, б.

На виході шинного підсилювача з прямим керуванням з'являється логічний нуль, якщо сигнал на керуючому вході $y_k = 1$, а вхідний сигнал $x = 0$; або логічна одиниця, якщо $y_k = 1$, $x = 1$. Він може також знаходитись у стані високого імпедансу, якщо керуючий сигнал $y_k = 0$. При цьому мікросхема буде від'єднана від шини даних (рис.52). У шинного підсилювача з інверсним керуванням сигнали проходять з входу на вихід, якщо керуючий сигнал $y_k = 0$ (рис.53).

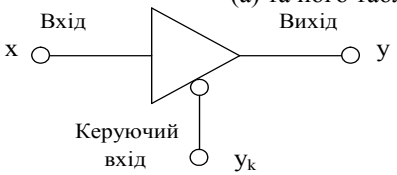


а)

y_k	x	y
1	0	0
1	1	1
0	$R_{xy} \to \infty$	

б)

Рис. 52. Умовне позначення шинного підсилювача з прямим керуванням (а) та його таблиця істинності (б)



а)

y_k	x	y
0	0	0
0	1	1
1	$R_{xy} \to \infty$	

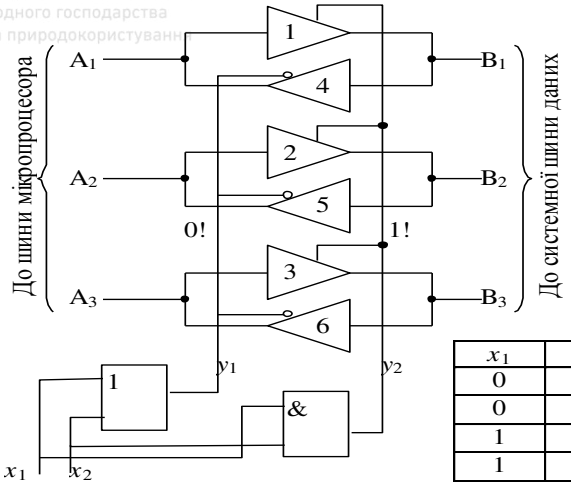
б)

Рис. 53. Умовне позначення шинного підсилювача з інверсним керуванням(а)та його таблиця істинності (б)

З енергетичної точки зору коефіцієнт підсилення схеми за потужністю дорівнює $k_s = 100000$і тому схема має назву підсилювача.

Схема багатовходового двонапрямленого шинного формувача та його таблиця істинності наведені на рис. 54 а, б.

Згіднно таблиці істинності при комбінації x_1=0, x_2=0; на виходах елементів “АБО” y_1=0 і “І” y_2=0. При цьому в активному стані будуть тристабільні елементи 4,5,6; а елементи 1,2,3 будуть у стані високого опору (Z→∞). Тому сигнали можуть передаватись від В до А, тобто від системної шини даних до МП. При x_1=1, x_2=1 на виходах елемента “АБО” y_1=1, і на виході елемента “І” y_2=1. В активному стані будуть елементи 1,2,3 і сигнали будуть передаватися від А до В, тобто від мікропроцесора у системну шину даних. При комбінаціях x_1=0, x_2=1 (y_1=1, y_2=0) або x_1=1, x_2=0 (y_1=1, y_2=0), елементи знаходяться у стані високого імпедансу і передачі сигналів не буде.



x_1	x_2	Y
0	0	B→A
0	1	∞
1	0	∞
1	1	A→B

а) б)

Рис. 54. Схема багатовходового двонапрямленого шинного формувача (а) та його таблиця істинності (б)

13.7. Інтерфейс пам’яті мікропроцесорних систем

З метою підвищення швидкодії та ефективності обміну інформацією між мікропроцесором і пам’яттю, її комірки діляться на області-сегменти, в кожному з яких є незмінним старший розряд. Їх комірки розміщуються в окремих мікросхемах, або зонах кристалу.

Мікросхеми пам’яті ППЗП (EP ROM) і ОЗП (RAM) під’єднаються до ШД та необхідної кількості ліній ША ($A_9 ... A_0$). Сигнали вибору мікросхем (кристалів) сегментів пам’яті формуються дешифратором DC (2:4), що забезпечує одночасний вибір тільки однієї мікросхеми з чотирьох, яка відповідає повному сегменту пам’яті (на схемі наведені дві мікросхеми). З таблиці станів дешифратора 24 слідує, що в будь-який момент часу тільки на одному з його чотирьох виходів $\overline{CS1}$, $\overline{CS2}, \overline{CS3}$ або $\overline{CS4}$ формується сигнал логічного “0”. При цьому дозвіл на декодування дає сигнал нульового рівня $\overline{G} = 0$, що надходить по адресній лінії $A15$.

Узагальнена схема інтерфейсу пам’яті має такий вигляд (рис.55а):

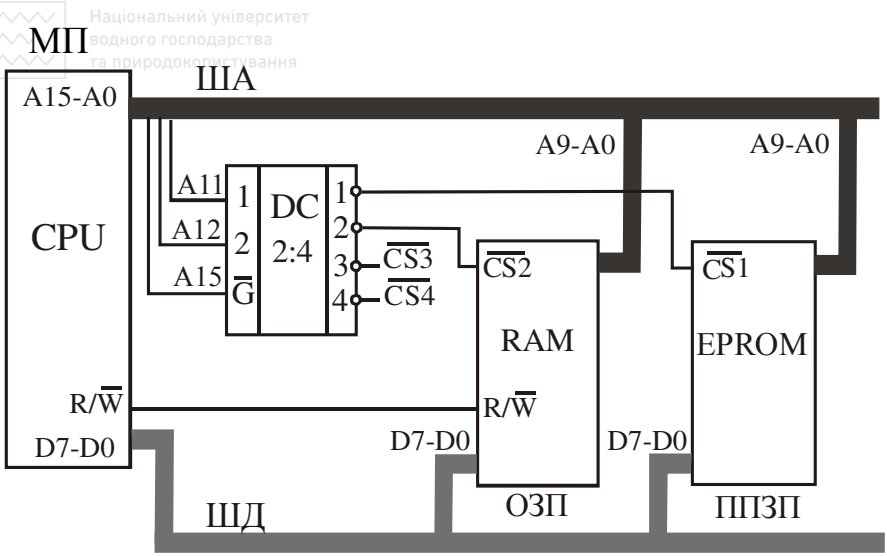


Рис. 55а. Схема інтерфейсу пам'яті мікропроцесорної системи: CPU – мікропроцесор; ША – шина адрес ($A_0 ... A_{15}$); ШД – шина даних ($D_0 ... D_7$); DC – дешифратор; ОЗП – оперативний запам'ятовуючий пристрій; ППЗП – перепрограмований постійний запам'ятовуючий пристрій.

Таблиця 24

Входи		Виходи			
A12	A11	$\overline{CS4}$	$\overline{CS3}$	$\overline{CS2}$	$\overline{CS1}$
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

Слово інформації зчитується з однієї з мікросхем постійної пам'яті при надходженні на їх входи $\overline{CS1}$ або $\overline{CS3}$ сигналів нульового рівня ($\overline{CS} = 0$). В оперативній пам'яті байт – слово інформації розміщується у 8 окремих мікросхемах, які створюють ряд і тому ОЗП, що зображені як окремі мікросхеми, створюють ряди, вибір яких здійснюється при надходженні на входи ОЗП сигналів $\overline{CS2}$ або $\overline{CS4}$ нульового рівня ($\overline{CS} = 0$). Така організація оперативної пам'яті дозволяє максимально використовувати адресний простір пам'яті і оптимально планувати схему системної плати. В ОЗП передбачається додатковий сигнал $R/\overline{W}$ - зчитування /запису, який керує направленням передачі байта.

Вибір комірки в кристалі ППЗП або ряду мікросхем в схемі ОЗП здійснюється дешифраторами, вбудованими в кристали ППЗП або змонтованими в блоках ОЗП. Для наведеної схеми інтерфейсу пам'яті формуються адреси і коди, які наведені в таблиці 25:

Таблиця 25

ША	A15	A14	A13	A12	A11	A10	A9..A0	H-код
Початок ПЗП	0	×	×	0	0	×	0…0	0000
Кінець ПЗП	0	×	×	0	0	×	1…1	07FF
Початок ОЗП	0	×	×	0	1	×	0…0	0800
Кінець ОЗП	0	×	×	0	1	×	1…1	0FFF

× – сигнал не використовується.

Мікросхеми ППЗП під'єднуються таким чином, що його комірки мають молодші адреси, які починаються з "0000" (в шістнадцятковій системі числення). Вони використовуються для розміщення програми монітора, яка починає працювати після вмикання мікропроцесорної системи (МПС), або після натиснення кнопки "СКИД", внаслідок чого формується сигнал "RESET" і CPU (central procesor unit) починає виконувати команду, що записана у комірці за адресою "0000" в шістнадцятковому коді Н.

Адреса в 16-ому коді	Використання адресного простору	Обсяг
0000 . . . . 07FF	ПЗП	2К
0800	ОЗП	2К
0FFF	СТЕК	
1000 . . . . FFFF	РЕЗЕРВ	62К

Рис.55 б. Карта пам'яті

Після визначення об'ємів постійної і оперативної пам'яті (табл. 25), виділяється резерв області пам'яті (на випадок можливих збільшень кількості їх адрес, а також для портів введення / виведення) і складається карта пам'яті, яка використовується при написанні програм. Приклад карти пам'яті наведений на рис. 55 б.

13.8. Контролер гнучких дисків зовнішніх накопичувачів FDC (floppy disk controller)

Це програмована мікросхема введення/виведення (рис. 56), яка використовується для керування зовнішніми накопичувачами.

Рис. 56. Типова мікросхема контролеру гнучкого диску

Дві лінії її адреси визначають чотири внутрішніх регістри (табл..26).

Таблиця 26

A1	A0	Регістри
0	0	Регістр даних (для передачі байта на диск або з диску)
0	1	Регістр доріжки (вибір доріжки)
1	0	Регістр сектору (вибір сектору)
1	1	Регістр стану (команди)

Передача інформації при запису відбувається таким чином. Програма завантажує регістр доріжки ($n_o = 80$) і регістр сектору ($n_c = 26$) відповідно з необхідною позицією на диску. FDC переміщує головку на вибрану доріжку, посилаючи імпульси на кроковий двигун, і очікує підходу вибраного сектора до головки. Потім головка опускається і генерується сигнал переривання. Програма завантажує в регістр даних байт, що буде записаний на поверхню магнітного диску. Цей байт перетворюється у послідовну форму і сигнали (імпульси) надходять в обмотку головки. Операція повторюється, поки не буде записаний весь блок даних.

Аналогічно проходить процес зчитування даних.

14. Інтерфейс введення/виведення даних

Пристрої введення/виведення (ПВВ) призначені для під'єднання МПС до периферійних пристроїв: клавіатури, дисплею, зовнішної пам'яті, принтера, АЦП, а також елементів керування виконавчими механізмами і т.п. Мікропроцесор використовує команду *IN* для введення або команду *OUT* для виведення даних за допомогою портів введення/виведення.

Коли МП виконує команду завантаження акумулятора безпосередньо з комірки зовнішнього пристрою (рис.57), лінія вибору A15 знаходиться в H-активному стані і на виході зчитування $\overline{RD}$ (L – активний вихід) з'являється низький потенціал ("0"), що приводить до активізації елемента захвату даних (схема "І"). Буфер виводу дає дозвіл і дані по лінії D(ШД) надходять в МП (CPU), де і розміщуються в акумуляторі.

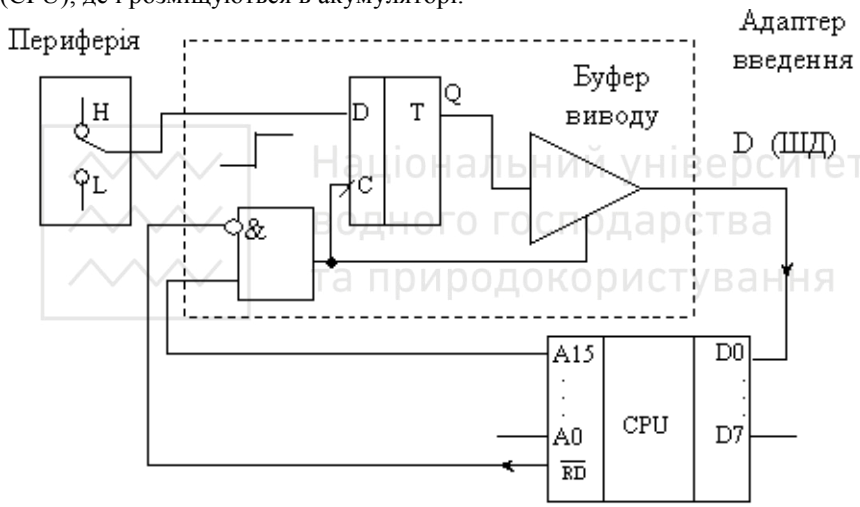


Рис. 57. Інтерфейс для введення окремого біта даних

При виведенні біта даних (рис.58) лінія A15 також знаходиться в H-активному стані і після того, як вихід записування $\overline{W}/R$ переходить в L-активний стан, відбувається активізація схеми "І". Біт інформації запам'ятовується в D-тригері і передається на периферійний пристрій.

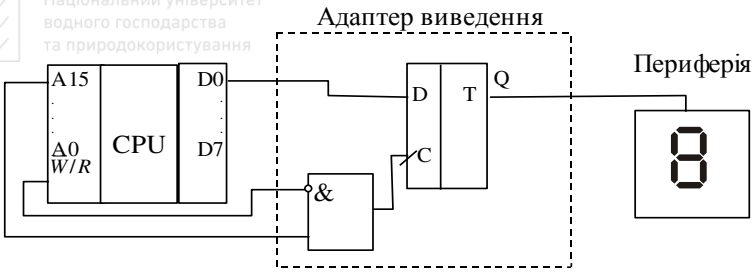
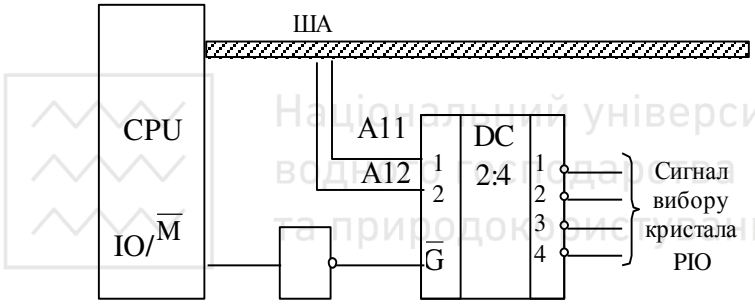
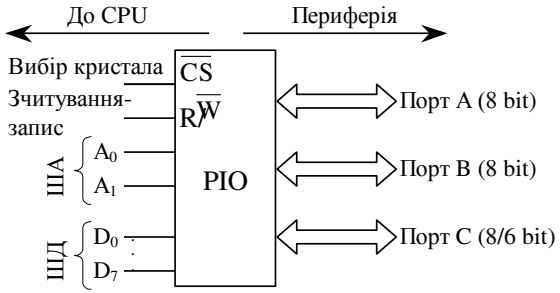


Рис. 58. Інтерфейс для виведення окремого біта даних

Розглянемо структурну схему паралельного інтерфейсу введення/виведення (рис. 59).



а)



б)

Рис. 59. Структурна схема паралельного інтерфейсу:
а) під'єднання дешифратора (DC);
б) типова схема паралельного ПВВ (PIO).

Під'єднання до шин є аналогічним до під'єднання ОЗП. Байт даних передається по ШД, а дві лінії ША забезпечують такі 4 адреси в мікросхемі РІО:

Таблиця 27

A1	A0	Порти, регістри
0	0	Порт А
0	1	Порт В
1	0	Порт С
1	1	Регістр управління

Регістр управління визначає напрямок портів, тобто програміст повинен розмістити керуючий байт в регістри управління (програмована МПС) до ініціювання передачі даних через порти.

При паралельному введенні (ПВ) забезпечуються такі режими роботи:

- введення аналогових сигналів з АЦП;
- введення дискретних сигналів через схему обробки;
- введення інформації з клавіатури.

Пристрої виведення (ПВВ) забезпечують виведення на табло або на дисплей, принтер та виконавчі механізми керуючих сигналів.

При передачі даних по одній лінії застосовують послідовний інтерфейс передачі, що забезпечується мікросхемою універсального асинхронного прийомопередавача (УАПП або UART) (рис.60). Такий принцип використовується при під'єднанні дисплея, окремих типів принтерів, а також в лініях зв'язку між комп'ютерами при створенні комп'ютерних мереж.

Під'єднання UART до МП є аналогічним під'єднанню РІО, плюс

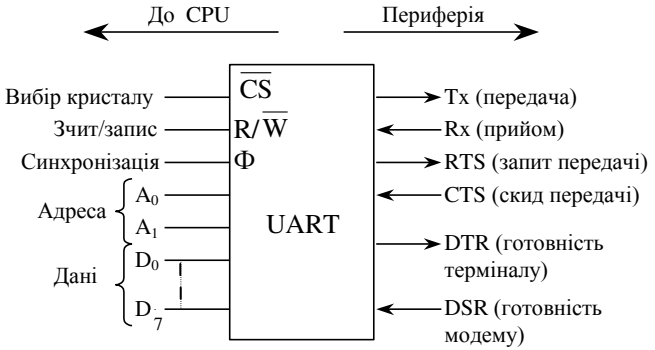


Рис. 60. Структурна схема послідовного інтерфейсу ПВВ

додатково подається синхронізуючий сигнал Ф для послідовної передачі 8 біт в кожному байті даних. Під'єднання зовнішнього периферійного пристрою (дисплею) здійснюється по лініям, що під'єднуються до виводів Tx, Rx. Разом з цим передбачено 4 звітуючих сигнали, які визначають, що

периферійний пристрій закінчив обробку інформації попереднього байта даних і готовий до передачі. В мікросхемі передбачені 4 адреси:

Таблиця 28

A_1	A_0	Порти, регістри
0	0	Tx (послідовний вихід)
0	1	Rx (послідовний вхід)
1	0	Регістр керування
1	1	Сигнал стану (символ переданий/прийнятий)

Слід зазначити, що при послідовному введенні/виведенні даних інформація виділяється стартовим (“0”) і стоповим (“1”) імпульсами. Всі схеми інтерфейсу, як і CPU, працюють з напругами ТТЛ-рівнів (“1” – +5В, “0” – 0В), але UART під’єднується до периферії за допомогою спеціального з’єднувача (RS-232C), в якому “1” – -12В, “0” – +12В і тому потрібна схема перетворення рівнів. Регістр керування задає кількість бітів даних (7) і стопових біт (1 або 2) та швидкість передачі інформації (600...9600) бод (біт/с). З метою завадозахищенності він додає до бітів інформації 1 біт парності (parity).

15. Керування введенням/виведенням даних

Застосовують такі способи керування процесами введення/виведення даних.

1. Спосіб послідовного опитування. Мікропроцесор опитує стан кожного порту і визначає, чи надходила в нього інформація або чи готовий він прийняти дані з метою їх наступного виведення. Про стан порту повідомляють однобітові сигнали (флаги). Спосіб є простим, але процес опитування повільний і мікропроцесор витрачає багато часу, оскільки кожного разу він перериває виконання основної програми, щоб опитати всі порти на випадок зміни їх станів.

2. Введення/виведення по перериванню. Цей спосіб вимагає більш складних апаратних та програмних засобів, але значно підвищує ефективність роботи мікропроцесора. Якщо порт є готовим до введення/виведення інформації, він передає сигнал спеціальному контролеру введення/виведення (КВВ) по шині управління. По адресній і керуючій шинам КВВ передає мікропроцесору адресу порта, в який надійшла інформація, а також адресу комірок ОЗП, в яких зберігається початок програми переривання для даного порту. Мікропроцесор завершує виконання поточної команди і записує вміст лічильника команд в стек. Він починає виконання підпрограми обслуговування операції введення /виведення та розміщення даних з портів в ОЗП, після чого повертається до виконання основної програми.

3. Способ прямого доступу до пам'яті (ПДП або DMA). При цьому способі введення/виведення здійснюється безпосередньо між пам'яттю та периферійними пристроями без участі мікропроцесора. Мікросхема введення/виведення посилає сигнал мікропроцесору: звільнити ШД, ША і дві лінії ШУ. Мікропроцесор переводить шинні підсилювачі у стан високого імпедансу (тобто відключається від шин) і підтверджує отримання сигналу. Після цього мікросхема введення/виведення адресує пам'ять і передає інформацію. Такий спосіб є найбільш ефективним з точки зору швидкодії і використовується при формуванні сигналів для кольорових дисплеїв.

Мікропроцесорна система може знаходитись або в системному стані і виконувати програму, або в стані введення/виведення. Перехід із системного стану в стан введення-виведення відбувається по сигналам *переривання* – "запитанням на переривання." При цьому система припиняє виконання системної програми і починає виконання програми введення/виведення.

Коли сигнал переривання надходить в мікропроцесор, то припиняється виконання системної програми, а вміст лічильника команд та загальних регістрів передається в стек. Після цього з пам'яті зчитується початкова адреса підпрограми введення/виведення та додаткові дані і пересилаються у лічильник команд та загальні регістри. Мікропроцесорна система переходить попередньо у режим очікування, а після надходження спеціального керуючого сигнала починається процес виконання програми введення-виведення. Після завершення операції введення/виведення вся інформація, що зв'язана з нею, зчитується з лічильника команд PC та загальних регістрів і записується в оперативну пам'ять. В лічильник та загальні регістри повертається інформація зі стеку і тільки після цього продовжується виконання перерваної програми.

Маскування – це процес ігнорування переривань під час введення-виведення. Масковане переривання може бути заборонене, а немасковане – система заборонити не може. В базовому мікропроцесорі немаскованим є переривання TRAP, всі інші переривання є маскованими.

Розглянемо схему механізму маскування і демаскування на такому прикладі (рис. 61). Якщо на вхід мікропроцесора надходить сигнал маскованого переривання RST, то перевіряється біт маски B_0 в регістрі стану. При $B_0 = 1$ сигнал залишається маскованим, переривання RST не визнається і мікропроцесорна система продовжує виконання програми в заданій послідовності. Але, якщо біт маски скинутий в нуль $B_0 = 0$, то додатково перевіряється індикатор дозволу переривання B_3. При $B_3 = 1$ переривання дозволено і визнається мікропроцесором, що фіксується появою на виході мікропроцесора сигнала $\overline{INTA}$ (або *INTE*, рис. 40). Виконання системної команди припиняється і система починає виконання програми введення/виведення.

Базовий 8-розрядний мікропроцесор містить 5 входів апаратного переривання:

– сигнал найвищого пріоритету *TRAP*, який є немаскованим перериванням і змушує мікропроцесор зберігати вміст лічильника команд в стеці;

– сигнали переривань більш низьких пріоритетів RST(Restart) ;

– переривання найнижчого пріоритету *INTR*(Interrupt). Певні біти регістра стану можуть бути змінені за допомогою команд *SIM* - встановити маску переривання (*Set interrupt mask*) або *RIM* - зчитати маску переривання (*Read interrupt mask*).

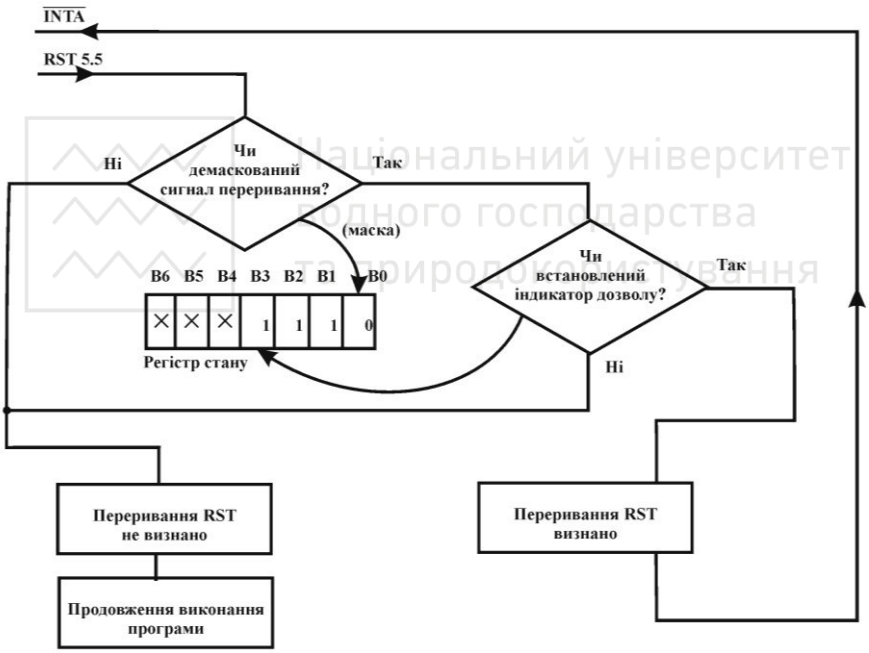


Рис. 61. Схема механізму маскування

16. Мова програмування асемблера

Мова асемблера (від англійського to assemble – компонувати) – це символічне представлення машинної мови. Кожна машинна команда позначається символом, що відповідає скороченій формі найменування даної команди англійською мовою. Наприклад:

Таблиця 29

Операція	Англійське найменування	Мнемонічне позначення
Записати у пам'ять	STORE	ST
Переслати	MOVE	MOV
Завантажити	LOAD	LD
Порівняти	COMPARE	CMP

і т. п. Мнемонічні позначення використовують також і для операндів та їх адрес. Для перекладу програми в машинний код використовують мовний транслятор. Підготовка програми на машинній мові за допомогою заміни символічних назв операцій на машинні коди, а символічних адрес на абсолютні або відносні адреси має назву асемблювання. Процес трансляції виконується за допомогою ЕОМ або вручну. Транслятор, який забезпечує переклад програми з мови асемблера в машинний код називають також асемблером (компілятором, від compile – збирати, склеювати). Завжди слід мати на увазі, що мова асемблера є апаратно-орієнтованою. Різниця в використанні асемблера на різних МПС виникає переважно у термінах команд та способах адресації операндів. І тому, при програмуванні на асемблері, необхідно чітко уявляти архітектуру конкретної МПС.

Програма мовою асемблера – це послідовність команд, що описують розв'язок поставленої задачі. Запис всіх команд ділиться на чотири частини – поля:

| мітка | | операція | | операнд | | коментар |.

Мітка – адреса команди, на яку посилаються в програмі під час її виконання.

Операція – це дія команди, поле операції містить її мнемонічний код.

Операнд – це дані, які обробляються, або їх адреси.

Коментар – це інформація для пояснення команди (її дій), яка не враховується при трансляції але часто буває важливою.

Початок програми прив'язується до конкретної адреси, відносно якої ведеться відлік відносних адрес всередині програми. Програми на асемблері складаються з використанням текстових редакторів.

Структурна схема алгоритму складання програми на асемблері має вигляд, зображений на рис. 62.

Спочатку складається вихідна програма, яку можна представити, наприклад, у вигляді такої таблиці:

Таблиця 30

Мітка	Операція	Операнд	Коментар
–	MVI	A, B4	Завантажити в А дані, що безпосередньо знаходяться за КОП:(B4)
	CMA		Інвертувати вміст А
	STA	2100	Розмістити вміст А у комірці пам’яті за адресою 2100
	HLT		Зупинити виконання програми

Рис. 62. Алгоритм складання програм

Після цього вручну або на ЕОМ переходять до об'єктної програми, в якій визначаються комірки пам'яті кожного коду операції та операнду, а мнемоніка КОП та операндів перекладається на машинну мову:

Таблиця 31

Адреса	Вміст	КОП.	Операнд	Коментар
2000 2001	3E B4	MVI	A,B4	Завантаження А
2002	2F	CMA		Інвертування А
2003 2004 2005	32 00 21	STA	2100	Розміщення вмісту А в комірці з адресою 2100
2006	76	HLT		Зупинка

Для записування програми на асемблері треба знати склад команд та архітектуру МПС. При цьому мають місце такі етапи:

1. Визначення та аналіз задачі;
2. Складання структурної схеми програми ;
3. Запис програми на асемблері;
4. Трансляція програми в код мікропроцесора ;
5. Запуск програми;
6. Оформлення документації.

17. Система команд базового мікропроцесора

Система команд – це набір команд, які може виконувати мікропроцесор певного типу. Команди різних мікропроцесорів можуть дещо відрізнятися. Більшість однобайтових, двобайтових і трибайтових команд складається з двох частин: операції і операнда. Операція вказує на дії (операції), які повинен виконувати мікропроцесор, наприклад: "додати", "відняти", "переслати" і т. д. Друга частина вказує на дані, які обробляються, або на їх адресу. Окремі однобайтові команди не мають операнду, наприклад: "заборонити переривання", "зупинити виконання програми" і т. д. В однобайтових командах старші розряди призначені для розміщення коду операції (КОП), а молодші – для коду регістрів (*R*).

В двобайтових командах перший байт використовується для розміщення КОП, а другий – для записування операнду у вигляді даних або адреси. 8 біт другого байту дозволяє записати найбільше число $N_{\max} = 255_{(10)}$. При необхідності введення чисел більших, ніж $N = 255$ (до $65535_{(10)}$) використовують трибайтові команди, в яких перший байт призначений для

розміщення КОП, а другий і третій байти – для розміщення шістнадцяткових двійкових чисел або шістнадцятирозрядної адреси комірки пам'яті. При написанні команд треба вірно вибирати спосіб адресації, який вимагає найменшого об'єму пам'яті, а також мінімального часу виконання команди.

Основні команди восьмирозрядного базового мікропроцесора умовно діляться на такі групи:

- команди пересилань даних;
- команди арифметичних дій;
- команди логічних операцій;
- команди переходів;
- спеціальні команди.

17.1. Команди пересилання даних

Однобайтова команда *MOV R*1, *R*2 означає: переслати в регістр *R*1, який є приймачем, дані з регістру *R*2, що є джерелом (рис. 63):

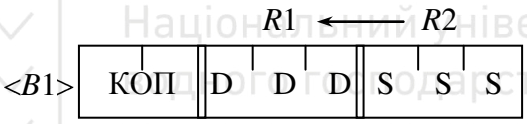


Рис. 63. Структура команди *MOV R*1, *R*2: *D* – біти регістра-приймача*R*1(destination або *DDD*); *S* – біти регістра-джерела *R*2 (sourse або *SSS*);*R*1, *R*2 – регістри загального призначення (*A*, *B*, *C*, *D*, *E*, *H*, *L*)

При пересиланні даних вміст джерела зберігається незмінним, а початковий вміст приймача змінюється результатом операції:

Однобайтові команди :

- ***MOV R, M*** означає: переслати в регістр *R* вміст комірки пам'яті *M*;
- ***MOV M, R*** означає: переслати в комірку пам'яті вміст регістру *R*.

На відміну від попередньої команди для її реалізації спочатку необхідно помістити в регістрову пару "*HL*" адресу комірки пам'яті *M*.

Двобайтова команда *MVI R, K* служить для розміщення в будь-якому регістрі загального призначення довільного числа *K*. Ця команда відрізняється від команди міжрегістрових пересилань (*MOV R*1, *R*2) тим, що дані надходять в регістр-приймач не з регістра-джерела, а із другого байту самої команди, в якому попередньо розміщують 8-розрядне двійкове число *K*. Буква *I* (Immediate) в кінці мнемонічного коду вказує на наявність числа *K*.

Команда *MVI M, K* є варіантом команди *MVI R*, *K* і дає можливість завантажувати комірку пам'яті *M* двійковим 8-розрядним числом. При цьому адреса комірки *M* повинна знаходитись в регістровій парі "*HL*".

Трибайтова команда *LXI P, K* на відміну від команди *MVI R, K* завантажує 16-розрядні числа в регістрові пари *P* (*BC*, *DE* або *HL*). В першому байті розміщують код операції, а в другий і третій байти записують по 8 розрядів 16-розрядного числа. При цьому в другий байт записують молодші розряди, а в третій – старші.

Трибайтова команда *LDA ADR* використовується для завантаження акумулятора (регістр *A*) вмістом комірки пам'яті *M*. При цьому адреса *ADR* комірки пам'яті *M* записується у другий та третій байти команди, а 1-ий байт призначений для коду операції.

Трибайтова команда *STA ADR* призначена для пересилання вмісту акумулятора в комірку пам'яті *M*, причому адреса комірки знаходиться у 2-му і 3-му байтах команди.

Двобайтова команда *IN N* забезпечує введення даних в акумулятор *A* із порту за номером *N* пристрою введення. По аналогії з номером комірки пам'яті *M*, його називають адресою *N*. В цій команді приймачем завжди є акумулятор *A*, і тому він не вказується в операнді. Перший байт команди показує КОП, а другий – номер (адресу) порту пристрою введення.

Двобайтова команда *OUT N* використовується для виведення даних із акумулятора *A* в порт пристрою виведення за адресою *N*. Джерелом даних в цій команді є акумулятор *A*, і тому він не вказується в операнді. Перший байт відводиться для КОП, а другий – для запису адреси порту виведення.

Під час виконання основної програми регістри загального призначення мікропроцесора *CP* завжди завантажені даними проміжних обчислень і перетворень. При переході до підпрограми **команда *PUSH***, як було наведено вище, пересилає вміст регістрів до комірок стекової пам'яті, а після виконання підпрограми по **команді *POP*** – відбувається повернення вмісту із комірок стекової пам'яті в ті самі регістри, з яких вміст було передано у стек.

17.2. Команди арифметичних операцій

Всі команди арифметичних операцій виконуються тільки через акумулятор *A*. При цьому один з операндів знаходиться в акумуляторі ще до виконання команди. Таким чином, додаючи два числа, необхідно одне з них розмістити в акумуляторі *A*, а друге можна завантажити в регістр *R*, комірку пам'яті *M* або записати в другий байт команди. Тільки після цього слід виконувати команду додавання. Після виконання команди сума автоматично розміщується в акумуляторі *A*, а число, яке було там, стирається.

Базовий 8-розрядний мікропроцесор виконує тільки команди додавання і віднімання, а операції множення і ділення здійснюються за особливими підпрограмами відповідно до правил, які були розглянуті у розділах "Двійкова арифметика" та "Двійкова арифметика у додатковому коді".

Однобайтова команда додавання з регістром *ADD R* призначена для додавання вмісту регістра *R* до вмісту акумулятора *A*, причому, сума знову записується в акумулятор: $R + A \to A$.

Однобайтова команда додавання з регістром та врахуванням ознак переносу *ADC R* виконує операцію арифметичного додавання і враховує знак переносу *C* (carry), отриманий після виконання команди, у вигляді зміни флагів регістру станів : $A + R + C \to A$.

Трибайтова команда додавання з пам'яттю *ADD M* виконує операцію арифметичного додавання числа в акумуляторі *A* з числом в комірці пам'яті *M*. При цьому у першому байті записаний КОП, у другому – старший, а у третьому – молодший байт регістрової пари *HL*, в яку занесена адреса комірки *M*: $A + M \to A$.

Крім команд додавання використовуються також команди віднімання, які не є необхідними, але значно полегшують процес програмування. Ці команди виконуються аналогічно командам додавання, але при відніманні в акумуляторі *A* розміщують зменшуване, а в регістрі *R*, комірці пам'яті *M*, або у другому байті команди повинен бути розміщений від'ємник.

Однобайтова команда віднімання з регістром *SUB R* виконує віднімання із вмісту акумулятора *A* числа, розташованого в регістрі *R*, а різниця знову записується в акумулятор: $A - R \to A$.

Однобайтова команда віднімання з регістром і переносом *SCB R*: $A - R - C \to A$.

Трибайтова команда віднімання з пам'яттю *SUB M*: $A - M \to A$.

Якщо різниця є негативною, то знаковий розряд регістра результату встановлюються в 1.

Слід зазначити, що всі операції віднімання виконуються мікропроцесором, як операція додавання зменшуваного до від'ємника у додатковому коді.

До арифметичних операцій відносяться також команди додатного приросту (інкремент) та від'ємного приросту (декремент) на "1" вмісту регістрів загального призначення або комірок пам'яті.

Однобайтова команда інкремент регістра *INR R* дає можливість збільшити на одиницю вміст регістра $R + 1 \to R$, а команда *INR M* – збільшити на одиницю вміст комірки пам'яті $M + 1 \to M$.

Однобайтова команда декремент регістра *DCR R* дає можливість зменшити на одиницю вміст регістра: $R - 1 \to R$, а команда *DCR M* – зменшити на одиницю вміст комірки пам'яті: $M - 1 \to M$. Адреса комірки *M* розміщується в регістрах *HL*.

Однобайтові команди *INX P* і *DCX P* є аналогічними командам інкремента і декремента стосовно до регістрових пар *P*. Команда *INX P* збільшує на одиницю вміст регістрової пари *P*, а команда *DCX P* зменшує його на одиницю.

17.3. Команди логічних операцій

Однобайтова команда *ANA R* виконує логічне множення (операцію *І*) над двома 8-розрядними двійковими числами, одне з яких знаходиться в акумуляторі *A*, а друге – в регістрі *R*. Логічна дія виконується над числами порозрядно. Можливі також варіанти команд: однобайтової *ANA M*, якщо друге число розташоване в комірці пам'яті *M*, або двобайтової *ANI K*, якщо друге число знаходиться у другому байті команди. Результат операції розміщується знову в акумуляторі.

Приклад виконання операції "*І*":

1-е число (*A*)	1001 0010
2-е число (*R*)	1100 1110
Результат "*І*" (*A*)	1000 0010

Однобайтова команда *ORA R* виконує логічне додавання (операцію АБО) над двома 8-розрядними двійковими числами, одне з яких знаходиться в акумуляторі *A*, а друге – в регістрі *R*.

Якщо друге число знаходиться у комірці пам'яті *M*, то попередньо у регістрову пару *HL* заноситься її адреса і використовується однобайтова команда *ORA M*. Двобайтова команда *ORI K* виконує операцію АБО над вмістом акумулятора *A* і членом *K*, записаним у другому байті команди.

Приклад виконання операції АБО:

1-е число (*A*)	1001 0010
2-е число (*R*)	1100 1110
Результат "*АБО*" (*A*)	1101 1110

Команди виключаючого АБО *XOR*: *XRA R*; *XRA M* і *XRI K* виконують операції над числами, розташованими в акумуляторі *A* і числами, які знаходяться відповідно у регістрі *R*, комірці пам'яті М або другому байті цієї команди (число *K*).

Приклад виконання операції виключаюче АБО:

1-е число (*A*)	1001 0011
2-е число (*R*)	1100 1110
Результат *XOR* (*A*)	0101 1101

Команда *CMA* виконує операцію логічного заперечення (інверсії) "*НІ*". При цьому всі одиниці у всіх розрядах числа змінюються на нулі, а всі нулі на одиниці, тобто число інвертується.

Команда *CMP R* порівнює числа в акумуляторі *A* і в регістрі *R*, команда *CMP M* – числа, які знаходяться в акумуляторі *A* та комірці пам'яті *M*, а по команді *CPI K* – число *K*, що записане у другому байті команди порівнюється з числом, розташованим в акумуляторі *A*.

Порівняння виконується за допомогою внутрішнього віднімання вмісту регістру *R*, комірки пам'яті *M* або числа *K* із вмісту акумулятора *A*. Результат визначається по стану флагів *Z* і *AC* регістру станів. Якщо числа є рівними, то флаг *Z* встановлюється в одиницю, а якщо ні, то тригер флага скидається в нуль. Якщо порівнювальне число більше, ніж число в

акумуляторі, то спрацьовує тригер флага *AC* і встановлюється в одиницю, а якщо менше, то тригер встановлюється в нуль. На відміну від команди віднімання *SUB* результат обчислень не розміщується в акумуляторі *A* і не змінює його вмісту, а впливає на регістр станів.

Команди зсуву застосовуються для виконання операцій множення, ділення та перетворень даних. Операція зсуву – це переміщення кожного розряду числа, записаного в акумуляторі, на один розряд вліво або вправо. В операціях зсуву бере участь тригер *C* регістру станів, який виконує функцію додаткового, "старшого розряду" акумулятора.

Команда зсуву на один розряд вліво *RLC* зображена на схемі (рис. 64):

Зі схеми зсуву слідує, що тригер *C* весь час дублює значення старшого розряду числа, що знаходиться в акумуляторі.

Приклад. Знайти кількість кроків програми зсуву *RLC* для

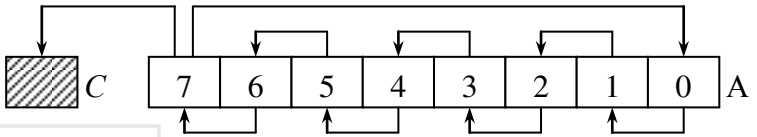


Рис. 64. Схема зсуву числа вліво при команді *RLC*

переміщення 5-го розряду регістру "*B*" в тригер *C* ("восьмий розряд").

Спочатку необхідно переслати вміст регістру *B* в акумулятор: *MOV A*, *B*. Щоб зсунути розряд з 5-го розряду у 8-ий необхідно виконати три команди *RLC*: $n = 8 - 5 = 3$.

Команда зсуву на один розряд вправо *RRC* зображена на схемі рис. 65.

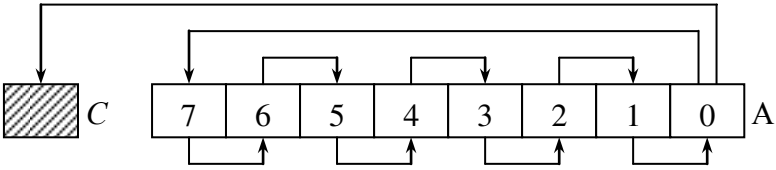


Рис. 65. Схема зсуву числа вправо при команді *RRC*

17.4. Команди переходу та спеціальні команди

По **трибайтовій команді *JMP ADR*** виконується безумовний перехід до виконання нової команди, адреса якої вказана у другому і третьому байтах команди. Для повернення до вихідної програми в кінці нової програми необхідно знову записати команду *JMP ADR* з указанням в ній тієї адреси, з якої почався перехід.

Трибайтова команда *CALL ADR* забезпечує безумовний перехід до виконання підпрограми, а **однобайтова команда *RET*** – безумовне повернення з підпрограми в основну програму. На відміну від команди *JMP ADR*, команда *CALL ADR* здійснює автоматичне повернення

після виконання підпрограми. Для цього адреса повернення розміщується у комірках пам'яті стеку, а в кінці підпрограми пишуть команду повернення *RET*.

Умовні переходи на відміну від безумовних залежать від стану тригерів (флагів) регістра станів, функціональні властивості яких розглядались вище. Адреси трибайтових команд умовних переходів записуються у 2-ому і 3-ому байтах команди. Якщо умова не виконується і тригер (флаг) регістра станів не встановлюється у стан логічної "1", то перехід не відбувається і продовжується виконання команд основної програми. Двійковий код тригерів (флагів) вказується в першому байті команди.

Команда умовного переходу *JNZ ADR* виконується при ненульовому результаті, якщо $Z = 0$, а команда *JZ ADR* виконується при нульовому результаті, якщо $Z = 1$.

Однобайтова команда *RST X* здійснює перехід до однієї з восьми спеціальних підпрограм режиму переривання, розташованих за фіксованою адресою *X*. Код команди надходить в мікропроцесор безпосередньо з периферійного пристрою. Команда *RST X* забезпечує терміновий перехід (або переривання) на іншу програму в аварійних ситуаціях.

Група спеціальних команд не здійснює дії над даними. Команди не мають операнду і містять тільки КОП. Більшість з них є службовими командами, які змінюють режим роботи мікропроцесора.

Команда *EI* – дозволяє переривання, а **команда *DI*** – забороняє переривання. Якщо програміст не поставив команду *EI*, то мікропроцесор ігнорує запитання на переривання і це може привести до небажаних наслідків. Якщо переривання програми недоцільне, то його заборону здійснюють за допомогою команди *DI*.

При команді *NOP* мікропроцесор не обробляє дані, але витрачає час 4-х машинних тактів на різноманітні перемикання, що створює необхідну затримку під час виконання програми. Якщо з'являється необхідність у введенні в програму допоміжної команди, то її розміщують взамін команди *NOP*.

Команда *HLT* зупиняє виконання чергової команди і переходить в режим зупинки. Щоб вивести мікропроцесор з цього стану, подається сигнал скиду або переривання.

18. Програми мікропроцесорних систем

18.1. Математичні програми

Розглянемо методику складання математичних програм на прикладі програми додавання вмісту 3-х послідовних комірок пам'яті з наступним занесенням суми в пам'ять. Для цього спочатку складається її структурна схема (рис. 66) і карта пам'яті (рис. 67), а потім об'єктна програма, що наведена в табл. 32.

Рис. 66. Структурна схема алгоритму програми

Пам'ять даних

Адреса	Вміст	Коментар
2010	0F	
2011	09	
2012	06	
2013		←сума

Програмна пам'ять

2020	21	к.1 завант. 2010 в H,L
2021	10	
2022	20	
2023	7E	к.2 завант. 1ч. в A
2024	23	к.3 збільш. HL
2025	86	к.4 додати 2-ге число
2026	23	к.5 збільш. HL
2027	86	к.6 додати 3-тє число
2028	23	к.7 збільш. HL
2029	77	к.8 помістити суму в M
202A	76	к.9 Стоп

Рис.67. Карта пам'яті

Таблиця 32

Адреса	Вміст	Операція	Операнд	Коментар
2020 2021 2022	21 10 20	LXI	H,2010	Завантажити адресу 2010 в пару H,L вказівника адреси
2023	7E	MOV	A,M	Завантажити число з комірки M 2010 в A
2024	23	INX	H	Збільшити H,L до 2011
2025	86	ADD	M	Додати друге число в комірці M 2011 до A
2026	23	INX	H	Збільшити H,L до 2012
2027	86	ADD	M	Додати третє число в комірці M 2012 до A
2028	23	INX	H	Збільшити H,L до 2013
2029	77	MOV	M,A	Помістити суму, що містить A, в комірку 2013
202A	76	HLT		Стоп

18.2. Підпрограми і стек

Великі програми спрощуються при застосуванні підпрограм для розв'язку задач, які мають повторювані дії. Завдяки цьому програми можна розділити на окремі модулі (секції).

Стек – це область оперативної пам'яті, яка призначена для зберігання адреси повернення, при виході з основної програми в підпрограму. Для звернення до підпрограми застосовується команда CALL.

Підпрограма – це частина основної програми, яка винесена окремим блоком. Як правило, підпрограми розміщуються після основної програми і викликаються з неї багаторазово.

При зверненні до підпрограми мікропроцесор посилає вміст програмного лічильника (PC) після команди CALL в стек. Цей вміст лічильника є адресою повернення до основної програми. Він повертається в програмний лічильник при виконанні команди повернення RET, якою завершується підпрограма (рис. 68).

Процедура обслуговування переривання ISR виконується, коли на шині управління з'являється сигнал переривання. Вона є аналогічною до команди CALL, але викликається активним апаратним сигналом переривання.

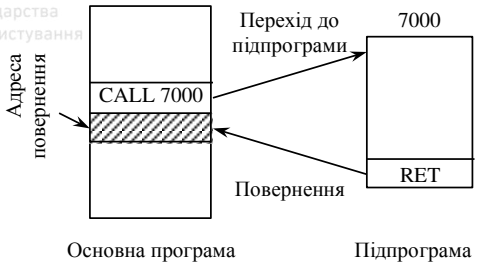


Рис. 68. Схема переходу до підпрограми та повернення в основну програму

Підпрограма процедури ISR є автономною програмою, і також закінчується командою повернення RET (рис. 69).

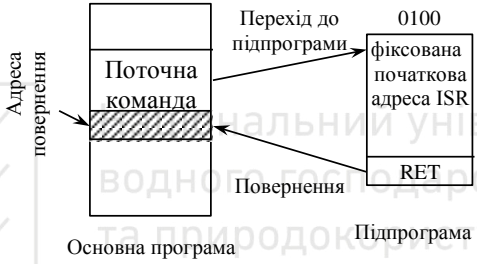


Рис. 69. Схема процедури обслуговування переривання

Команди CALL і RET взаємодіють із вказівником стеку автоматично. Разом з тим є команди, що дозволяють використовувати стек згідно до потреб поставленої задачі. Це команда PUSH (зберегти регістри в стеку) і POP (відновити регістри зі стеку). Під час дії кожної з цих команд відбувається автоматична зміна вказівника стеку.

Розглянемо механізм роботи стеку у 8-розрядного мікропроцесора (табл. 33). Вказівник стеку в МП – це 16-розрядний регістр, який містить адресу останньої використаної комірки стеку (вершина стеку). В початку основної програми розміщують команду LD SP, яка завантажує вказівник стеку. Після виконання команди CALL адреса повернення 2073 розміщується у другій комірці стеку (спочатку молодший байт 73, потім – старший – 20), причому вказівник стеку зменшується на 2.

Щоб зберегти вміст робочих регістрів A,B,C та регістру стану при виконанні підпрограми та відновити його в кінці підпрограми або після обробки переривання, використовують команди PUSH і POP. Команда PUSH відсилає дані з регістрів в стек, а команда POP відновлює значення регістрів із стеку.

Таблиця 33

Основна програма	Пам'ять	Коментар
2000	LD SP 6000	Завантажити у вказівник стеку адресу 6000
⋮	⋮	⋮
2070	CALL 3000	Викликати підпрограму за адресою 3000
2073		
⋮	⋮	⋮
	JMP 2000	Перейти у початок програми
Підпрограма		
3000	PUSH AF	Зберегти A і флаги F в стеку
	PUSH BC	Зберегти B і C в стеку
⋮	⋮	⋮
	POP BC	Відновити B і C зі стеку
	POP AF	Відновити A і флаги F зі стеку
	RET	Повернення
Стек		
	Вміст C	Після PUSH BC
	Вміст B	
	Вміст F	Після PUSH AF
	Вміст A	
	73	Після CALL
	20	
6000		← Вказівник стеку

18.3. Відгалуження програм

Використання знаків прийняття рішення призводить до внутрішнього відгалуження програми. При цьому алгоритм порівняння двох величин, які представлені відповідними числами N_1 і N_2, має таку структуру:

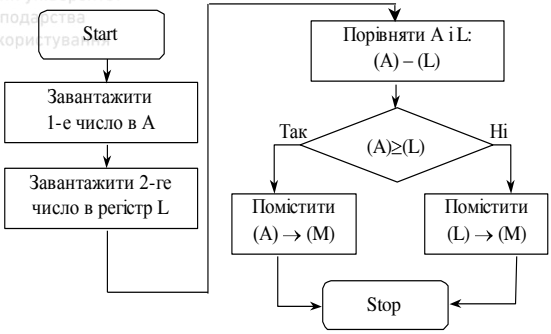


Рис. 70. Блок – схема алгоритму відгалуження програм

Розглянемо принцип побудови програми відгалуження на ассемблері:

Таблиця 34

Мітка	Мнемоніка	Операнд	Коментар
	MVI	A, N_1	Завантажити 1-е число (N_1) в А
	MVI	L, N_2	Завантажити 2-ге число (N_2) в L
	CMP	L	Порівняти (А) і (L), встановити флаг CY=1 при (А)<(L)
	JC	STORE L	Перейти до STORE L, якщо CY=1; якщо ні – продовжити послідовно
	STA	2040	Помістити (А) в комірку пам'яті за адресою 2040
	HLT		Зупинити МП
STORE L	MOV	A,L	Передати (L) в аккумулятор
	STA	2040	Помістити вміст (А) в комірку пам'яті за адресою 2040
	HLT		Зупинити МП

Таким чином відгалуження здійснюється командою "JC", згідно якої приймається рішення відповідно положенням флагів регістру станів мікропроцесора. Мітка STORE L – це символічна адреса при команді переходу.

18.4. Циклічні програми

Цикли в програмі є ефективним методом її скорочення при виконанні задач, що повторюються. Наприклад, алгоритм циклічної програми

розміщення ряду чисел (від 0 до 8) послідовно в пам'яті за адресами від $2040_{(16)}$ до $2048_{(16)}$ має таку структуру:

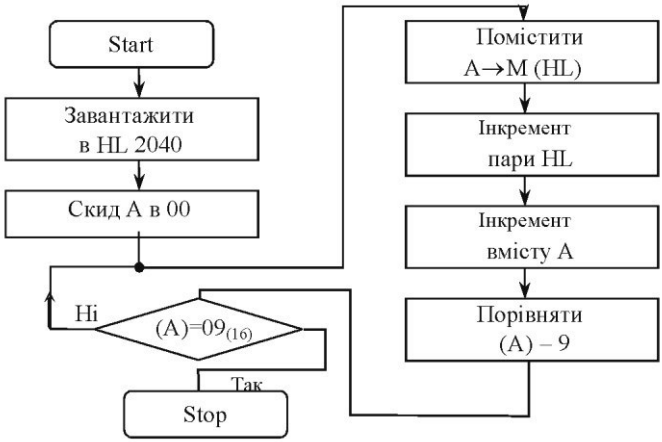


Рис. 71. Структурна схема алгоритму циклічної програми

Розглянемо версію циклічної програми базового мікропроцесора:

Таблиця 35

Мітка	Мнемоніка	Операнд	Коментар
	LXI	H,2040	Завантажити 2040 в пару HL (вказівник адреси)
	XRA	A	Скид акумулятора в 00, тобто A⊕A=00 (виключаюче АБО)
LOOP	MOV	M,A	Помістити вміст A в комірку пам'яті за адресою, вказаною в HL
	INX	H	Інкрементувати пару HL
	INR	A	Інкрементувати вміст A
	CPI	09	Порівняти A=$09_{(16)}$ Якщо так, то встановлюється флаг Z=1
	JNZ	LOOP	Перехід до LOOP, якщо Z=0 |A-09|>0;а якщо ні - продовження програми
	HLT		Зупинити МП

Команда CPI порівнює вміст A з константою 09. Якщо A містить числа між 0 та 8,- то флаг нуля скинутий у нуль. Але якщо A = 9, то флаг встановлюється в "1". JNZ перевіряє флаг нуля. Якщо він скинутий в 0, то| A-09|>0. При цьому здійснюється перехід на символічну адресу LOOP. Якщо

флаг встановлений в 1, то A-09=0, - програма виходить з цикла і виконується послідовна команда HLT.

19. Мікропроцесорна система Z80

Подальшим розвитком базової мікропроцесорної системи є восьмирозрядна універсальна система *Z80*, що розроблена фірмою *Zilog*.

Мікропроцесор системи *Z80* виконаний у вигляді великої інтегральної схеми (*BIC*) по *n*-канальній МДН-технології з кремнієвими затворами. Його програмне забезпечення є сумісним з програмним забезпеченням *Intel 8080* (*8085*), але команди мають дещо іншу мнемоніку. Порівняно з базовим мікропроцесором програмне забезпечення мікропроцесора *Z80* має такі переваги:

– розширений набір команд;
– спрощені схеми інтерфейсу;
– більша кількість рівнів переривань;
– динамічне ОЗП з регенерацією вмісту динамічної памті.

Напруга живлення мікропроцесора дорівнює $U_H = +5\ B$, максимальна споживана потужність $P_{max} = 1{,}1\ Bm$ і частота однофазної синхронізації $f_{ti} = 4{,}0 \div 6{,}0\ МГц$. Мікропроцесор має 16 адресних виводів $A_0 \ldots A_{15}$, що дає можливість за допомогою адресної шини (ША) оперувати з адресним простором пам‘яті 64 *К*. Двонапрямлена шина даних (ШД) під’єднується до 8 виводів $D_0 \ldots D_7$. Виводи шини управління (ШУ) призначені для сигналів управління системою і мікропроцесором СР. Робота мікропроцесора *Z80* здійснюється при подачі на нього тактових імпульсів. Команда виконується за кілька машиниих циклів, кожний з яких містить кілька тактів (тактових періодів). Наприклад цикл виконання операції *LD* (*HL*), *n* складається з 3-х машинних циклів. Під час першого машинного циклу зчитується КОП, у другому машинному циклі з ППЗП зчитується значення *n* і в третьому машинному циклі значення *n* записується в ОЗП за адресою, що вказана у регістрі *HL*.

До основних машинних циклів *Z80* відносяться:

– зчитування КОП;
– зчитування або запис даних в пам’ять;
– зчитування або запис даних введення/виведення;
– запитання або підтвердження доступу до шини;
– запитання переривання з маскуванням, підтвердження дозволу на переривання;
– запитання або підтвердження переривання без маскування;
– відміна зупинки.

Загальна кількість команд *CPU Z80* дорівнює 158, причому до їх складу входять двійкові коди всіх 78 команд мікропроцесора *8080.*

Внутрішні регістри *CPU Z80* A, B, C, D, E, H, L і F є аналогічними відповідним регістрам базового мікропроцесора, але в ньому існує допоміжний (альтернативний) набір регістрів A′, B′, C′, D′, E′, H′, L′ і другий регістр стану F′.

	Основні регістри		Допоміжні регістри	
	Акумулятор	Флаг	Акумулятор	Флаг
Регістри загального призначення	A	F	A′	F′
	B	C	B′	C′
	D	E	D′	E′
	H	L	H′	L′

Регістри спеціального призначення	Вектор переривань I	Регенерація пам‘яті R
	Індексний регістр IX	
	Індексний регістр IY	
	Вказівник стеку SP	
	Програмний лічильник PC	

Наявність допоміжного набору регістрів значно спрощує виклик підпрограм (*CALL ADR*), або процедуру обслуговування переривання (*ISR*). Це обумовлено можливістю використання альтернативних регістрів A′, B′, C′, D′, E′, H′, L′ і F′ замість зберігання вмісту регістрів основної програми в стеку.

Обмін вмісту всіх регістрів здійснюють такі дві команди: *EXX* – обмін між BC, DE і HL; *EX AF, AF′* – обмін між AF і AF′.

Восьмибітовий регістр вектора переривання “*I*” використовують для локалізації початкової адреси *ISR*, коли *CPU* працює в одному з трьох режимів переривань: *RESET, NMI* (немасковане переривання) або *INT*.

Регістр регенерації “*R*” активується командами *RFSH* в інтервалах між робочими командами і забезпечує регенерацію динамічних ОЗП, які під‘єднуються до шин *CPU*. Після кожної команди відбувається інкремент регістра “*R*”.

Два 16-бітних індексних регістра IX і IY забезпечують в командах індексний режим адресації.

В мікропроцесорі *Z80* використовують такі способи адресації:

- регістровий *LD A, C* (завантажити в аккумулятор вміст регістра С);
- прямий *LD A, (5020)* (завантажити в аккумулятор вміст комірки пам‘яті 5020);
- регістровий посередній *LD A, (HL)* (завантажити в аккумулятор вміст комірки пам‘яті, адреса якої знаходиться в HL);
- безпосередній *LD A,* $N_{(16)}$ (завантажити в аккумулятор 8-бітне число $N_{(16)}$);
- індексний *LD A, (IX+2)* (завантажити в аккумулятор вміст комірки пам‘яті, адреса якої дорівнює збільшеному на 2 вмісту регістра IX).

В командах переходу передбачені два режими адресації:

- абсолютна *JP NZ, 0500* (перейти, якщо не нуль, за адресою 0500);
- відносна *JP Z, -9* (перейти, якщо нуль, за відносною адресою – 9, тобто на 9 байт назад від адреси наступної команди).

Розглянемо методику складання програм для мікропроцесора *Z80* на прикладі програми передачі двійкового коду числа *FF* в 10 комірок пам‘яті, починаючи з адреси 3000 (табл. 36).

Таблиця 36

Мітка	Операція	Операнд	Коментар
CONT	LD	A, FF	*Завантажити в А код числа FF*
	LD	D, (10)	Завантажити в D вміст лічильника циклу (10)
	LD	BC, 3000	Завантажити в BC код 3000
	LD	(BC), (A)	Записати вміст А в пам‘ять (посередньо через BC)
	INC	BC	Інкремент BC
	DEC	D	Декремент лічильника цикла в D
	JP	NZ, CONT	Повторити цикл 10 разів
	HLT		Зупинити МП

Для сумісної роботи з *CPU Z80* застосовується мікросхеми паралельного введення/виведення (*Z80 PIO*).

Структурна схема мікроконтролера з мікропроцесором Z80, паралельним інтерфейсом введення/виведення, динамічною оперативною пам’ятю обсягом 48 К та багатоканальним АЦП наведена на рис. 72. Вхідні сигнали через АЦП надходять в порти PIO 1. Звідси вони зчитуються і обробляються CPU згідно програми, записаної в ROM 0, ROM 1. Сформовані ним керуючі сигнали записуються в RAM або динамічну оперативну пам’ять і через порти А, В інтерфейсу PIO 2 надходять на периферійні пристрої. Лінії зчитування/запису $R/\overline{W}$ на структурній схемі (рис. 72) не наведені.

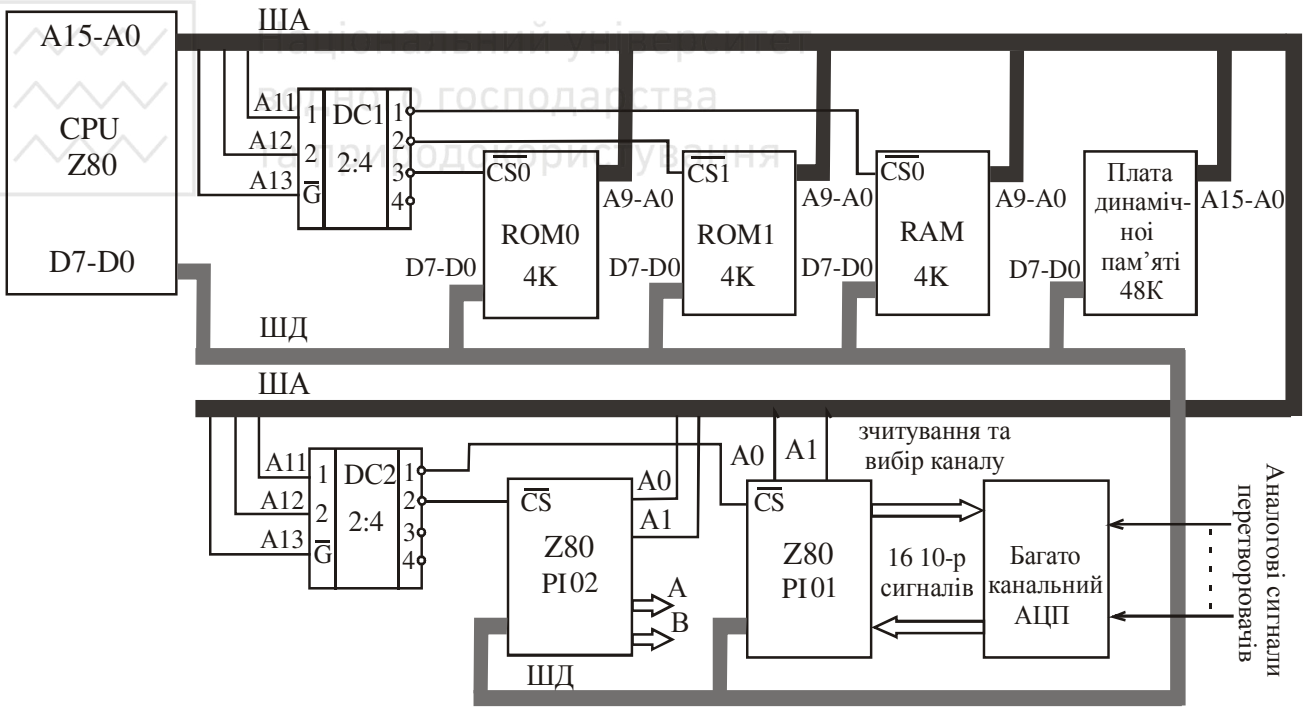


Рис. 72. Структурна схема мікроконтролера з мікропроцесором Z80 та динамічною оперативною пам'ятю

20. Шістнадцятирозрядні мікропроцесори, їх структура і програмування

Ці мікропроцесори в порівнянні з восьмирозрядними мікропроцесорами мають значні переваги: більшу швидкодію, розширену систему команд, збільшений обсяг адресної пам'яті (1 Мбайт і більше порівняно з 64 Кбайтами восьмирозрядних мікропроцесорів), можливість застосування співпроцесорів (допоміжних процесорів) завдяки чому оптимізується виконання програм. Базові шістнадцятирозрядні мікропроцесори (Intel 8086) складаються з операційного пристрою (EU) та шинного інтерфейсу (BIU) (рис.73). Вони виконують всі операції восьмирозрядних мікропроцесорів, а також операції множення і ділення, операції зсуву (логічного, арифметичного і циклічного на задане число розрядів) та ряд інших операції. 16-розрядні шини мікропроцесорів мультиплексуються. Сигнал ALE (дозвіл фіксації адреси) вказує на наявність адреси на шині. Адресна шина має 20 ліній, що забезпечує адресацію пам'яті 1 Мбайт.

Операційний пристрій містить такі основні компоненти центрального процесора СР: арифметично-логічний пристрій (АЛП), регістри тимчасового зберігання, регістр станів, регістр команд, пристрій керування та вісім регістрів загального призначення.

В шістнадцятирозрядних мікропроцесорах застосовують *сегментну* організацію пам'яті. Її використання надає можливість адресувати значно більший масив пам'яті за рахунок 20-розрядної шини даних ($N_{(20)}=2^{20}$=1 М > $N_{(16)}= 2^{16}$ = 64 К). При цьому для формування 20-розрядної адреси використовують дві 16-розрядних *логічних адреси.* Перша логічна адреса доповнюється чотирма нулями праворуч і є *початковою адресою* області пам'яті, що має назву *сегменту.* Друга логічна адреса визначає так зване *зміщення* в сегменті - відстань від початку сегмента до адресованої комірки. Якщо зміщення дорівнює $0000_{(16)}$, то адресується перша комірка сегменту, а якщо $FFFF_{(16)}$, то адресується його остання комірка.

Таким чином, фізична 20-розрядна адреса комірки пам'яті, формується з адреси сегменту A_{seg}, що додається зі зміщенням на чотири розряди ліворуч та виконавчої адреси A_{ex}. Це еквівалентно множенню адреси сегменту на величину 2^4 =16. Тому отримана *20-розрядна фізична адреса* дорівнює:

$$A_F = 16A_{seg} + A_{ex}.$$

Змінюючи значення логічних адрес A_{seg} і A_{ex} можна адресувати будь-яку комірку пам'яті із загального масиву ємністю $N_{(20)}$ = 1 М.

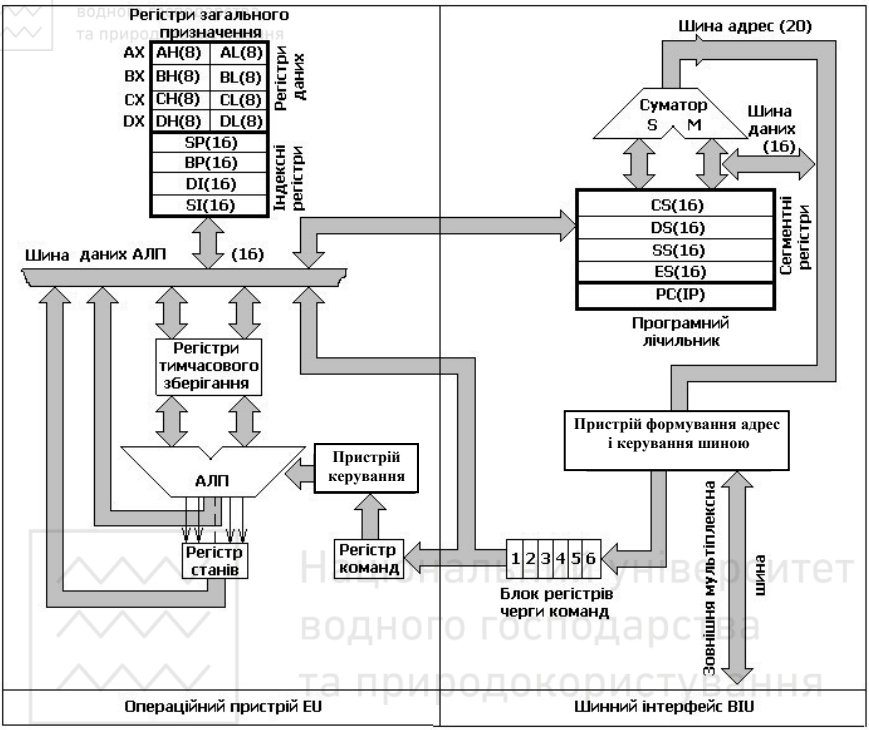


Рис.73. Структурна схема шістнадцятирозрядного базового мікропроцесора

Блок регістрів загального призначення складається з чотирьох 16-розрядних регістрів даних AX, BX, CX і DX. Крім цих регістрів, аналогічних регістрам 8-розрядного мікропроцесора, додатково передбачені чотири 16-розрядних індексних регістри SP, BP, DI і SI. При цьому регістр SP є вказівником стеку, а індексні регістри BP (вказівник бази), DI (індекс приймача) та SI (індекс джерела) використовують при формуванні адрес. В операційному пристрої EU виконуються команди обробки даних та формуються сигнали керування мікропроцесором.

Шинний інтерфейс BIU містить програмний лічильник PC (або вказівник команд IP), сегментні регістри: коду CS, даних DS, стеку SS та додатковий сегмент ES. До його складу входить також суматор SM, пристрій формування адрес та керування шиною, а також блок регістрів черги команд. Шинний інтерфейс формує адреси комірок пам'яті і забезпечує виклик команд та їх зберігання до початку виконання. В пам'яті мікропроцесорної системи обсягом 1 Мбайт виділені окремі області – *сегменти* з обсягом пам'яті по 64 Кбайт кожна, адреси яких послідовно зростають. Для визначення початкових адрес цих сегментів залежно від

інформації (команди, дані, стек) відповідно використовують *сегментні регістри* CS, DS, SS або ES.

Пристрій формування адрес та керування шиною з'єднаний із зовнішньою мультиплексною шиною адрес/даних і забезпечує розподіл сигналів та автоматично заповнює чергу команд *блоку регістрів черги команд* (рис.73). Цей блок забезпечує можливість накопичення команд обсягом до 6 байт. Завдяки цьому команди з блоку регістрів черги команд шинного інтерфейсу BIU подаються в операційний пристрій EU і виконуються без затримки (за рахунок економії часу на вибір команд із пам'яті). Тобто процеси вибору команд із пам'яті та їх реалізації суміщені в часі. Робота з чергою команд має назву конвейєрізації.

Суматор SM додає вміст одного із сегментних регістрів з адресою, яка є вмістом одного з регістрів загального призначення BX, BP, SI або DI (рис.73).

Для утворення 20-бітних адрес, як згадувалося вище, вміст 16-розрядного сегментного регістра зсувається ліворуч на 4 розряди і потім до нього додається зміщення (16-розрядна адреса в операнді команди), яке розміщено в одному з регістрів загального призначення (BX, BP, SI, або DI) (рис.74 а, б). Отримана фізична адреса видається на адресну шину.

а) сегментна пам’ять; б) обчислення фізичної адреси даних

Рис.74. Утворення 20-бітних адрес з використанням сегментних регістрів

Система команд базується на обробці значень даних, які знаходяться в регістрах загального призначення. Перші чотири регістри є основними регістрами перетворення даних і допускають доступ до їх байтів. Вони використовуються як акумулятори, але, разом з тим, виконують спеціальні функції. Наприклад, BX є базовим регістром при обчисленні адрес даних в пам'яті, а DX використовується для передачі даних у порти введення-виведення, вказівник бази BP використовують як джерело 16-бітної бази для

обчислювання адреси пам'яті. Індексні регістри джерела SI і приймача DI містять зміщення, які приймають участь у формуванні пам'яті. Розглянемо, як вибирається команда з пам'яті і як для неї зчитується 16-бітний елемент даних у режимі прямої індексної адресації (рис.75).

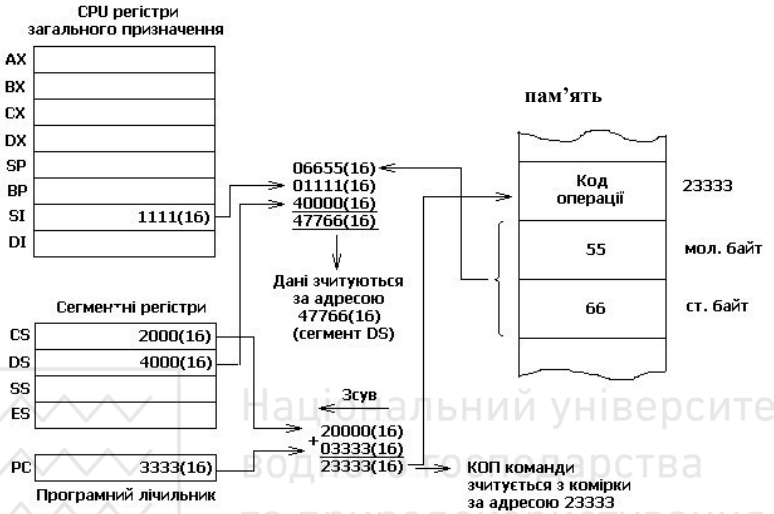


Рис.75. Приклад обчислення адреси пам'яті

В програмному лічильнику команд PC формується адреса команди 3333 (16), до якої додається вміст сегменту коду CS 2000 (16), попередньо зсунутий на 4 розряди вліво:

$$\begin{array}{r} 20000_{(16)} \\ +\ \underline{03333_{(16)}} \\ 23333_{(16)} \end{array}$$

Отримане число вказує адресу комірки, з якої зчитується код операції 6655 (16). До нього додається вміст індексу джерела S1 1111 (16) та зсунутий на 4 розряди вліво вміст сегмента даних DS 4000 (16). При цьому отримують адресу комірки сегменту даних, з якої беруться дані:

$$\begin{array}{r} 06655_{(16)} \\ +\ 01111_{(16)} \\ \underline{40000_{(16)}} \\ 47766_{(16)} \end{array}$$

Як згадувалось вище, порівняно з 8-розрядним базовим мікропроцесором список команд передбачає виконання ряду додаткових операцій. Основними з них є:

- команда множення вмісту акумулятора A на вміст регістра R або комірки пам'яті M: MUL R, MULM;

- команда ділення вмісту акумулятора A на вміст регістра R або комірки пам'яті M: DIV R, DIV M;

- знакове множення: IMUL R, IMUL M;

- знакове ділення: IDIV R, IDIV M;

- зсув вправо з лічильником в регістрі CL: SHR;

- зсув вліво з лічильником в регістрі CL: SHL;

- обмін вмісту регістрів з акумулятором: XCHG AX,(R);

- зміна знаку: NEG.

В командах використовуються наступні режими адресації:

безпосередня MOV AX,29 ;	Завантажити число 29 в AX;
регістрова MOV AX,BX ;	Завантажити в AX вміст BX;
пряма MOV AX, (1040) ;	Завантажити в AX вміст комірки пам'яті за адресою 1040;
посередня з базовим регістром MOV AX, (BX) ;	Завантажити в AX вміст комірки пам'яті, адреса якої знаходиться в BX;
посередня адресація з індексним регістром MOV AX, (SI) ;	Завантажити в AX вміст комірки пам'яті, адреса якої знаходиться в SI;
посередня з базовим і індексним регістрами (замість BX можна вказувати BP, а замість SI- регістр DI) MOV AX, (BX) (SI) ;	Завантажити в AX вміст комірки пам'яті, адреса якої дорівнює сумі BX і SI;
посередня з базовим та індексним регістрами плюс зміщення MOV AX, 2100 (BX) (SI);	Завантажити в AX вміст комірки пам'яті, адреса якої дорівнює сумі BX, SI, та 2100;
відносна JMP – 23 ;	Перейти (безумовно) до PC-23, тобто 23 байта назад.

Є також команди переходу та виклику зі звичайною абсолютною адресою, наприклад :

JMP 8000 ; Перейти (безумовно) на 8000.

Розглянемо приклад програмування 16-розрядного мікропроцесора :

Операція	Операнд	Коментар
MOV MOV	AX, 1000 ; DS, AX ;	Задати початкову адресу 1000 сегмента даних ;
IN	AL, 2 ;	Ввести байт із порта з адресою 2 ;
MOV	1000, AX ;	Зберегти в сегменті даних (адреса пам'яті дорівнює 1000) ;
OUT	3, AL ;	Вивести в порт з адресою 3 ;
HLT		Зупинка .

Програма зчитує байт вхідного порту в регістр AL, зберігає це значення в пам'яті, а далі видає введений байт в вихідний порт.

21. Архітектура шістнадцятирозрядних мікропроцесорних систем

Структура системи, виконаної на базовому 16-розрядному мікропроцесорі, наведена на рис.76. Вона складається з центрального процесора ЦП (CPU), робота якого синхронізується генератором тактових імпульсів ГТІ (fr=5 МГц); арифметичного співпроцесора АСП; співпроцесора введення/виведення ПВВ; контролера шини керування КШ, а також двох 8-розрядних прийомопередавачів 2×ПП, виходи яких підключені до 16-розрядної шини даних (D15…D0), і трьох 8-розрядних буферів 3х БФ, виходи яких підключені до 20-розрядної адресної шини (A19…A0).

АСП з високою точністю виконує операції над числами з фіксованими та плаваючими комами і над складними математичними функціями. ПВВ оптимізує операції, введення-виведення даних. При сумісній роботі з цими співпроцесорами ЦП працює у ***максимальному режимі*** внаслідок розширення шини керування за допомогою КШ.

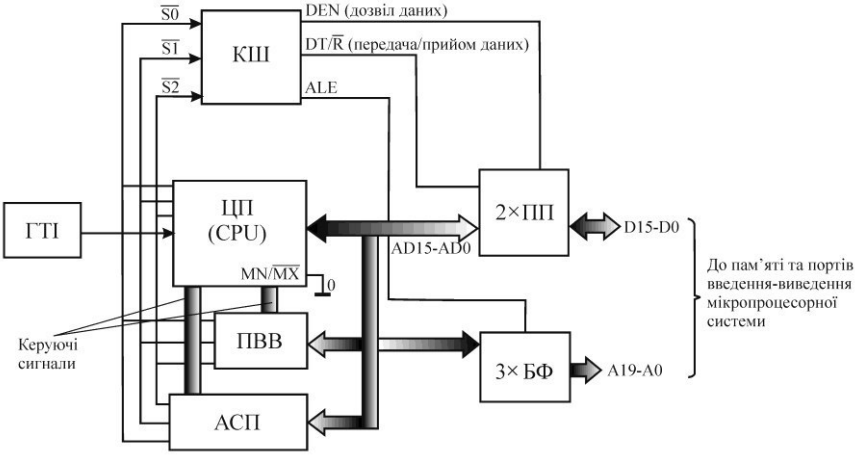


Рис.76. Структурна схема 16-розрядної мікропроцесорної системи

На входи КШ з ЦП надходять сигнали низького рівня $\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$, які утворюють слово стану, а на його виходах формуються сигнали: DEN (дозвіл даних), DT/$\overline{R}$ (передача/прийом даних) - для керування двонапрямленим шинним формувачем; ALE (adress leth enable) (наявність адреси на мультіплексованій шині) - для керування фіксатором адрес та інші.

Співпроцесор АСП контролює команди ЦП і визначає, чи повинен він виконувати замість нього певні операції над числами. Співпроцесор ПВВ запускається керуючим сигналом з ЦП. Його робота керується програмою, записаною в область пам'яті, яка доступна як для ЦП, так і для ПВВ. Ця область пам'яті має назву "***поштова скринька***". Програма ЦП завантажує в неї код операції введення/виведення та послідовність команд ПВВ.

Співпроцесор виконує команди (кількість яких біля 50) та перериває ЦП після закінчення процесу введення/виведення. ПВВ має два канали і може керувати передачею двох потоків даних. При обміні даними за допомогою прямого доступу до пам'яті без участі ЦП (ПДП або DMA) співпроцесор ПВВ дає команду ЦП звільнити шини і генерує сигнал запиту DMA. Після цього ЦП припиняє свою роботу і переводить мультиплексовану шину адрес/даних та лінії сигналів керування у високоімпедансний стан. Він виробляє сигнал підтвердження захоплення шин HLDA, який надходить на вхід ПВВ і співпроцесор бере на себе функції керування обміну даними. Після закінчення обміну даними ПВВ посилає на вхід ЦП сигнал HOLD,

внаслідок чого ЦП формує сигнал HLDA, відновлюючи керування шиною, і продовжує роботу за програмою.

При перериваннях вміст регістрів PC(IP), CS та регістру станів ЦП (рис. 73) пересилається в стек і виконується підпрограма переривань, після чого відбувається повернення системи до основної програми.

За допомогою двох 8-бітних прийомопередавачів 2х ПП та трьох 8-бітних буферів 3х БФ, центральний процесор, співпроцесори і контролер підключаються через демультиплексовану шину даних (D0...D15) та адресну шину (A0…A19) до пам’яті та портів введення/виведення мікропроцесорної системи.

Пам'ять 16-розрядної мікропроцесорної системи організована як послідовність суміжних байтів, які можуть утворювати двобайтові слова (молодшим байтам відповідають менші адреси). Звернення до мегабайтного адресного простору можливе як за байтовими, так і за двобайтовими словами. Для цього пам'ять розділена між двома банками ємністю по 512 Кбайт кожен, для адресації яких використано лінії A1…A19 (рис.77). Банк з парними адресами підключений до молодших (D7…D0), а банк з непарними адресами до старших (D15…D8) розрядів шини даних. Адресна лінія A0 для адресації пам'яті не застосовується, а використовується для передачі сигналу вибору банка з молодшими байтами. Сигнал BHE (byte high enable) надходить з ЦП і вибирає старші байти, а також дозволяє звертатися до банка зі старшими байтами, або обох банків.

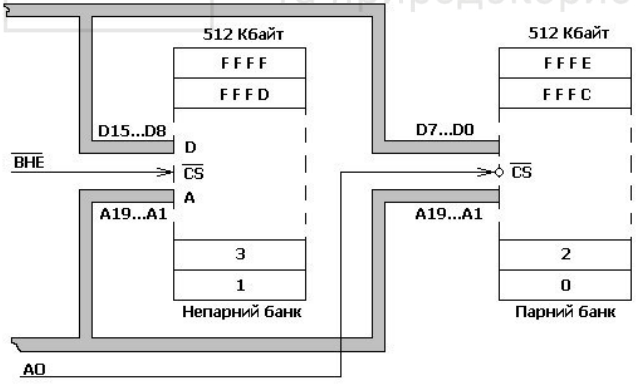


Рис.77. Організація пам’яті 16-розрядної мікропроцесорної системи

Завдяки такому керуванню можна звертатися до байтів з парними і непарними адресами, а слова можуть мати парні і непарні адреси. Таблиця істинності для сигналів BHE і A0 має такий вигляд:

Таблиця 37

$\overline{BHE}$	A0	Байт, що вибирається
0	0	Обидва байти (передача слова)
0	1	Старший байт (непарна адреса)
1	0	Молодший байт (парна адреса)

22. Операційні системи, їх призначення і структура

Програмне забезпечення мікропроцесорних систем завжди містить операційну систему – набір програм, що забезпечує їх функціонування.

Операційна система має два види програм: монітор і сервісний комплект. Монітор – це набір керуючих програм, які забезпечують зв'язок між апаратними засобами мікропроцесорної системи, програмами та користувачем. Він містить постійні резидентні програми та змінні, що розміщуються у зовнішніх накопичувачах, наприклад дисках, і створюють дискову операційну систему (DOS). Резидентні програми монітора завжди містять початковий завантажувач, програмне керування клавіатурою (при її наявності), а також можуть мати абсолютний завантажувач та відладчик.

Початковий завантажувач встановлює мікропроцесорні системи у вихідний стан при включенні живлення, а також при їх поверненні до початку виконання системної програми. Програмоване керування клавіатурою застосовується при наявності матриці з однополюсними нефіксованими кнопочними контактами, які у нормальному стані є розімкненими. При цьому позиція кожної клавіши визначається і змінюється програмним способом.

Абсолютний завантажувач – це програма, яка забезпечує введення із зовнішних накопичувачів в основну пам'ять системи трансльованих програм. Відладчик дозволяє перевіряти і змінювати вміст комірок пам'яті та регістрів мікропроцесора. Відладчик дає можливість виконувати покрокове виконання програми для визначення точок, в яких стан мікропроцесора відрізняється від заданого.

Сервісні програми виконуються під керуванням монітора для реалізації дій системного рівня. Вони діляться на допоміжні, які використовують для розробки прикладних програм, та програми для обслуги файлів. Допоміжні програми містять редактори тексту і транслятори. За допомогою редакторів тексту вносяться зміни та доповнення в команди. Транслятори – це спеціальні команди, що замінюють мнеономічний код його двійковим еквівалентом. Внаслідок машинної трансляції мікропроцесорні системи отримують об'єктні програми, що завантажуються в основну пам'ять. Їх важливою функцією є також роздрук програм, перевірка та визначення помилок. Існують два типи команд трансляторів: компілятори та інтерпретатори. При цьому компілятори забезпечують переведення у двійковий код всієї програми в цілому і створюють об'єктну програму для

завантаження у пам'ять мікропроцесорної системи певного типу. Інтерпретатор послідовно транслює кожен оператор вихідної програми у відповідний оператор об'єктної програми, тобто діє за принципом „рядок за рядком". Завдяки цьому стає можливою компоновка складної програми з окремих модулів і тому програмування є більш перспективним. Програми для обслуговування файлів містять програми копіювання , переміщення та анулювання (знищення) записаного тексту. Мікроконтролери (мікроЕОМ) можуть завантажуватись програмами з персональних комп'ютерів (ПК).

Операційні системи OS, що обслуговують 8-розрядні мікропроцесорні системи з дисковими накопичувачами, мають три програмні рівні:

- ядро, яке обробляє команди користувача, керує периферійними пристроями і пересилає файли між резидентною і зовнішньою пам'яттю системи;
- допоміжні програми – утиліти: редактор для створення нових програмних файлів, а також компілятори та інтерпретатори;
- прикладні програми.

Після включення живлення або сигналу скидання, програма початкового завантажувача зчитує програму операційної системи з диску в оперативну пам'ять мікропроцесорної системи і передає їй керування. Операційна система OS виводить на екран стимул (наприклад, символ „A>", якщо робочим є дисковий накопичувач A). Після цього користувач вводить команди, які забезпечують виконання необхідних програм. Отримані результати виводяться на екран дисплея або принтер.

Редактор операційної системи ED (рис.78) призначений для створення текстових файлів. Такі файли містять вихідні програми мовою асемблера або мовою високого рівня.

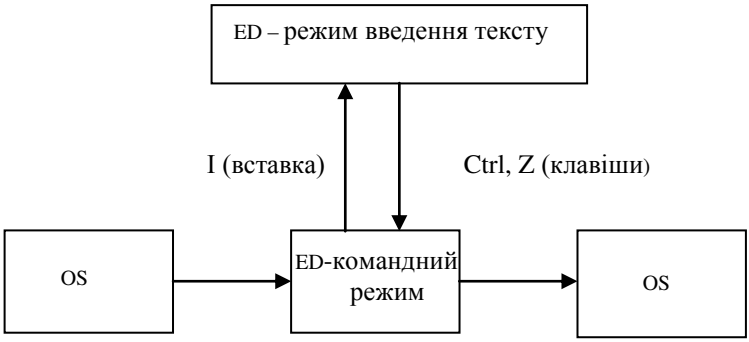


Рис. 78. Режими роботи редактора ED: OS – операційна система; ED – програма-редактор; I – завдання режиму вставки (введення тексту); Ctrl,Z – відміна режиму вставки.

Основні команди редактора ED наведені у таблиці 38 :

Таблиця 38

№ п/п	Команда	Коментар
1	E	Вихід на диск та зберігання файлу
2	H	Вихід на диск, зберігання файлу та повторний виклик E
3	O	Повернення до оригінального файлу
4	Q	Вихід без змін та повернення до OS
5	A	Додання файлу з диску
6	B	Переміщення вказівника рядка у початок файлу
7	P	Вказівник сторінки
8	L	Просування вказівника рядка по файлу
9	T	Вказівник рядків
10	K	Стирання рядків
11	S	Заміна низок рядків

Розглянемо використання цих команд на прикладі корегування текстового файлу, який містить програму мовою асемблера базового мікропроцесора.

1. Виклик ED здійснює команда: A>ED FILE NAME ASM (Enter). Після цього редактор готовий до роботи і виводить на екран дисплея, або на принтер свій стимул – символ *.
2. Додання попереднього файлу: *A# (Enter). Символ номера означає передачу з диску у пам'ять мікропроцесорної системи всього файлу.
3. Виведення на екран дисплея, або на принтер поточного файлу: *BOP (Enter). Команда B переміщує вказівник рядка на початок файлу, а OP виводить на дисплей першу екранну сторінку (при завданні 1P виводиться на екран друга сторінка).
4. Перехід до того рядка, де необхідно внести зміни: *B8L (Enter). Вказівник переміщується від початку файла на 8 рядків наперед, тобто на рядок 9.
5. Стирання поточного і наступного рядків : *2K (Enter). Стираються рядки 9 і 10.
6. Завдання режиму вставки (введення нового тексту) : *I (Enter). Введення нових рядків тексту, наприклад асемблерних команд.
7. Відміна режиму вставки: Ctrl, Z (Enter). Операція здійснюється одночасно з натисканням клавіш Ctrl (CONTROL) і Z з переходом в командний режим редактора.
8. Зберігання скоригованого файлу на диску: *E (Enter). Файл передається на диск і керування повертається до OS.

В подібних операційних системах переважно використовується мова асемблера базового мікропроцесора ASM COM. Для керування програмами, написаними мовою асемблера, використовують директиви:

- ORG (origin - початок) - директива завдання початкової адреси програми;
- EQU (equare - прирівняти) – директива присвоєння адрес регістрам мікроконтролерів, їх бітам та лініям портів;
- DB (define byte – визначити байт) – директива для резервування комірок пам'яті та розміщення в них байтів даних;
- END (end - кінець) – повідомлення про кінець програми.

Асемблер повідомлює про синтаксичні помилки (невірна мнемоніка, невизначенні мітки у вихідних файлах і т.п.). До повторного передавання файла всі помилки виправляються за допомогою редактора і формується 16-вий файл машинного коду.

Для відлагодження програм в операційній системі передбачений також динамічний засіб (DDT), за допомогою якого можна працювати з певними файлами робочої програми при її виконанні.

23. Однокристальні восьмирозрядні мікроконтролери

Високопродуктивні однокристальні мікроконтролери використовуються в материнських платах комп'ютерів, блоках керування двигунами та ходовим обладнанням машин, стільникових телефонах нового покоління, супутникових навігаційних системах та інших сучасних технологіях.

В однокристальних мікроконтролерах системи команд переважно орієнтовані на забезпечення процесів регулювання та керування і передбачена можливість розширення пам'яті та інформаційного простору введення/виведення.

23.1. Однокристальні мікроконтролери (ОМК) з CISC-архітектурою

Розглянемо структуру та функціонування ОМК з нейманівською архітектурою на прикладі однокристального мікроконтролера фірми Intel MCS-51(рис.79).

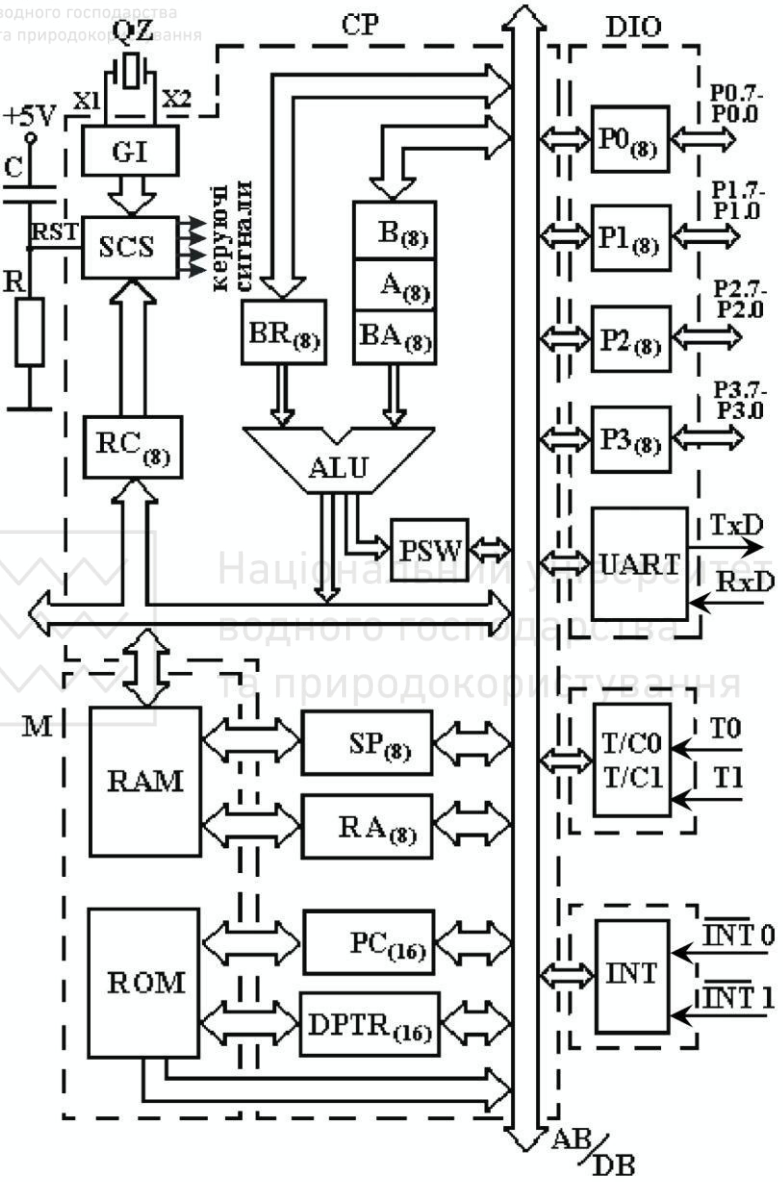


Рис.79. Структурна схема однокристального мікроконтролера MCS-51 з CISC-архітектурою

Його структурна схема містить мікропроцесорне ядро СР; пам'ять М (програмну ROM і даних RAM); 8-розрядні програмовані порти введення/виведення DIO: чотири паралельних порта $P0_{(8)}$-$P3_{(8)}$ та один послідовний порт UART, а також два 16-розрядні програмовані таймери /лічильники T/C0, T/C1 і систему переривань з п'ятьма векторами та двома рівнями пріоритетів INT.

Операційна частина мікропроцесорного ядра СР містить 8-розрядний арифметико-логічний пристрій ALU, два акумулятори $A_{(8)}$ і $B_{(8)}$, буферні регістри $BA_{(8)}$ і $BR_{(8)}$, регістр стану процесора PSW (Processor State Word), вказівник стеку $SP_{(8)}$, 8-розрядний регістр адреси $RA_{(8)}$, лічильник команд $PC_{(16)}$, регістр-вказівник даних DPTR (Date Pointer Register). Мікропроцесор MCS-51 виконує арифметичні операції додавання, віднімання, множення, ділення та логічні операції І, АБО, НІ, ВИКЛЮЧАЮЧЕ АБО, а також операції *зсуву і скидання*. В акумуляторі $A_{(8)}$ знаходиться один з операндів і в ньому розміщуються результати виконання операцій. Записування і зчитування даних з оперативної пам'яті RAM, а також команди TEST(перевірка), INC(інкремент), DEC(декремент) виконуються без участі акумулятора. Акумулятор В використовується як акумулятор лише під час виконання операцій множення і ділення, а в інших випадках його використовують як регістр загального призначення.

В регістр стану процесора PSW записується інформація про стан ALU під час виконання програми (табл.39).

Таблиця 39

Формат слова стану процесора

Біт	Позначення	Функціональне призначення флагів
0	P	Парність
1	–	Резерв
2	OV	Переповнення
3	RSO	Вказівник банку робочих регістрів: 00-банк 0 ; 01- банк 1 ; 10- банк 2 ; 11- банк 3.
4	RSI	
5	FO	Флаг користувача
6	AC	Додаткове перенесення
7	C	Перенесення

Флаг OV встановлюється при додаванні та відніманні, якщо розрядність результату перевищує 7 біт і старший біт не може вважатись знаковим. Він набуває значення логічної "1" при множенні, якщо добуток перевищує $OFF_{(16)}$, а також при діленні на "0".

Вміст лічильника команд PC(Program Counter) визначає адресу комірки внутрішньої пам'яті програм ROM. Регістр DPTR(Data Pointer Register) використовується як два 8-розрядні регістри, або один 16-розрядний регістр, а також як вказівник адреси при посередній адресації та при адресації таблиць. Вказівник стеку $SP_{(8)}$ призначений для адресації стека, який є

частиною внутрішньої пам'яті даних RAM. Його вміст інкрементується перед записом даних в стек за допомогою команд PUSH і CALL і декрементується за командами POP і RET (передінкремент/ постдекремент). Регістр адреси $RA_{(8)}$ є програмнедоступним і використовується для завантаження адрес при пошуку комірки в пам'яті даних RAM.

Керуюча частина мікропроцесорного ядра CP містить 8-розрядний регістр команд $RC_{(8)}$ та систему керування і синхронізації SCS, на яку надходять сигнали з генератора тактових імпульсів GI, стабілізованого кварцовим резонатором QZ.

Код команди, зчитаної з ROM, записується в регістр $RC_{(8)}$ і далі надходить на SCS, яка містить дешифратор команд, програмовану логічну матрицю та логіку керування, що створює керуючі сигнали. До складу SCS входить також внутрішній формувач синхроімпульсів, що формує внутрішні сигнали синхронізації машинних циклів та сигнал дозволу фіксації адреси ALE, і регістр керування споживанням (на структурній схемі не наведені). При подачі напруги живлення +5В, - на диференціюючому колі CR виникає сигнал загального скидання RST, після чого мікроконтролер починає працювати .

Постійна внутрішня або резидентна пам'ять програм ROM виконана у вигляді ПЗП, програмованих маскою, основним елементом яких є матриця елементів пам'яті з перемичками (діодними або транзисторними) на перетинах рядків і стовпців в точках з логічною "1". У тих точках матриці, де має бути логічний "0",- перемички відсутні. Занесення інформації в масочну пам'ять називається прошиванням. Інколи MCS-51 мають перепрограмовані запам'ятовуючі пристрої з ультрафіолетовим стиранням EP ROM. Пам'ять програм має 16-розрядну адресну шину, що дозволяє збільшити обсяг пам'яті з 4 Кбайт до 64 Кбайт за допомогою плати розширення. Молодші адреси пам'яті призначені для обслуговування переривань INT0, INT1 та забезпечення початку роботи ОМК після його скидання у нульовий стан.

Оперативна внутрішня або резидентна пам'ять даних RAM складається з двох областей. Область з адресами даних $0_{(16)} \div 7F_{(16)}$ має інформаційну ємність 128 байт, причому перші 32 байти організовані в чотири банка регістрів загального призначення (банк „0" ÷ банк „3"). Кожний банк містить 8 регістрів R0÷R7. Одночасно програмно-доступним є тільки один банк регістрів, номер якого містять 3-ий і 4-ий біти слова стану програми PSW (табл.39). За адресами $20_{(16)} \div 2F_{(16)}$ розташовані комірки, що містять біти з прямою адресацією, тобто мають індивідуальні адреси для кожного біту. За їх допомогою можна звертатись до будь-якого з бітів при використанні *команд операцій з бітами* (CLR bit - скидання біту, SETB bit - встановлення біту). Решта комірок цієї області використовується як звичайна оперативна пам'ять для розміщення стеку та записування/зчитування даних.

Друга область оперативної пам'яті RAM з адресами $80_{(16)} \div FF_{(16)}$ відведена для бітів адресного простору *регістрів спеціального призначення* SFR (Special Function Registers). До них відносяться акумулятори A* і B*; регістр слова стану програми PSW*; регістр–вказівник стеку SP; старший DPH і молодший DPL байти регістра-вказівника DPTR; порти паралельного введення/виведення P0*; P1*;P2*;P3*; регістр керування SCON* та буфер SBUF послідовного порту введення/виведення UART; регістр режимів таймерів/лічильників TMOD, регістр управління-статуса таймерів TCON, старший TH0 і молодший TL0 байти таймера/лічильника T/C0; старший TH1 і молодший TL1 байти таймера/лічильника T/C1; регістри дозволу IE* і пріоритетів IP* системи переривань INT, а також регістр керування споживанням PCON (регістри, що позначені зірочкою, припускають адресацію окремих бітів). Регістри SCON, SBUF входять до складу послідовного порту; TMOD, TCON – до складу таймерів/лічильників; IE, IP – до складу системи переривань, а PCON – до складу системи керування SCS і тому на структурній схемі MCS-51 (рис. 77) вони не наведені.

Порти паралельного введення-виведення P0÷P3 забезпечують побайтний обмін інформацією мікроконтролера з периферійними пристроями по 32 лініях введення-виведення. Кожна лінія порту містить регістр-фіксатор, два буферних регістри та вихідний транзисторний каскад. Будь-яку лінію портів можна використовувати для введення або виведення інформації незалежно від інших ліній. Щоб використовувати лінію порту для введення інформації, в D-тригер регістра-фіксатора треба записати логічну "1", яка закриває МДН-транзистор вихідного каскаду. P_0 - двонапрямлений 8-розрядний порт, призначений для введення/виведення інформації. При роботі із зовнішньою пам'яттю спочатку по лініях порту видається її адреса і тільки після цього здійснюється передача або прийом даних. 8-розрядний порт P1 також призначений для введення/виведення інформації і є квазідвонапрямленим, тобто перед її введенням у всі розряди регістра-фіксатора SFR потрібно записати логічні "1". При цьому кожен розряд порту може бути запрограмований як на введення, так і на виведення інформації, незалежно від стану інших розрядів. P2 – це квазідвонапрямлений 8-розрядний порт, аналогічний P1. Його виводи використовуються також для передачі 16-розрядної адресної інформації при зверненні до зовнішньої пам'яті. Порт P3 разом із звичайним введенням-виведенням інформації виконує також альтернативні функції обміну інформацією з приймачем послідовного порту UART, таймерами і системою переривань.

Послідовний порт UART призначений для послідовного обміну даними і може працювати як регістр зсуву, або як універсальний асинхронний прийомопередавач із фіксованою або змінною швидкістю обміну і забезпечує дуплексний обмін. Послідовний порт програмується на один з чотирьох режимів: "0","1","2","3" за допомогою запису керуючого слова у вищезгаданий регістр SCON(Serial Port Control). В режимі "0" послідовний

порт є 8-розрядним регістром зсуву. Байт інформації передається і приймається через лінію RxD, а по лінії TxD при цьому передаються синхронізуючі імпульси. В режимі “1” послідовний порт працює як 8-розрядний універсальний асинхронний прийомопередавач зі змінною швидкістю обміну, яка визначається часом переповнення таймера/лічильника. В режимах “2” і “3” послідовний порт є 9-розрядним універсальним прийомопередавачем з фіксованою (для режиму “2”) та змінною (для режиму “3”) швидкістю обміну. В режимі “2” швидкість обміну визначається тільки резонансною частотою кварцового резонатора f_Q, діленою на 32 або 64, а в режимі “3” швидкість обміну задається таймером.

Два 16-розрядних таймери/лічильники T/C0, T/C1 мають чотири режими роботи. В режимі “0” обидва таймери/лічильники працюють як 8-розрядний лічильник з переддільником частоти на 32. Відміна режиму “1” в тому, що таймери/лічильники перетворюються на пристрій з 16-розрядним регістром. В режимі “2” T/C0, T/C1 працюють як 8-розрядний лічильник молодшого байту таймерного регістру, який автоматично перезавантажується його старшим байтом після переповнення (скиду із одиничного в нульовий стан). В режимі “3” T/C1 заблокований і зберігає незмінним свій попередній вміст, а T/C0 працює як незалежний пристрій (два 8-розрядних таймери або лічильник).

Система переривань призначена для реагування на зовнішні та внутрішні переривання. При зовнішніх перериваннях нульовий потенціал або задній фронт (зріз) імпульсу з’являється на лініях $\overline{\text{INT0}}$, $\overline{\text{INT1}}$. До внутрішніх переривань відносяться переповнення таймерів/лічильників або завершення послідовного обміну. Переривання можуть також бути викликані або відмінені програмним шляхом. Як згадувалось вище, кожне з джерел переривань має один з двох рівнів пріоритету, - високий або низький. Схема формування вектора переривань формує адреси підпрограм обслуговування переривань (табл.40).

Таблиця 40

Вектори переривання

Джерело переривання	Вектор переривання
Зовнішнє переривання INT0	00 $03_{(16)}$
Таймер / лічильник T/C0	00 $0B_{(16)}$
Зовнішнє переривання INT1	00 $13_{(16)}$
Таймер / лічильник T/C1	00 $1B_{(16)}$
Послідовний порт	00 $23_{(16)}$

Система команд MCS-51 містить 111 команд, що діляться на команди пересилання, арифметичні команди, логічні команди, *команди операцій з бітами* та команди переходів. 94 команди мають формат з двох байт <B1;B2> і виконуються від одного до двох циклів (тривалість циклу $T_ц$=1 мкс при

тактовій частоті f_{TI} =12 МГц). В мікроконтролері використовують пряму, безпосередню та посередню адресації. Ознакою посередньої адресації є знак @ перед одним з трьох регістрів: R_0, R_1 або DPTR, з яких беруться адреси.

23.2.Однокристальні мікроконтролери з RISC-архітектурою

Аналіз програм однокристальних мікроконтролерів з CISC-архітектурою показує, що в 80 % випадків використовується тільки 20 % загальної кількості команд , причому більше 70 % площі кристалу займає дешифратор команд. Тому досить часто доцільно використовувати мікроконтролери з RISC-архітектурою, яка надає можливість скоротити кількість команд , привести їх до одного формату, зменшити площу кристалу та збільшити швидкодію мікроконтролера. Разом з тим , інколи більш повна система команд мікроконтролерів з CISC-архітектурою дає виграш і скорочує час виконання певних програм.

Найбільш перспективними мікроконтролерами з RISC-архітектурою є багатофункціональні мікроконтролери фірми Atmel , об'єднані загальною маркою AVR.

Вони всі містять мікропроцесорне ядро CP, енергонезалежну постійну FLASH-пам'ять програм і EEP-пам'ять даних, оперативну пам'ять даних DATE RAM, восьмирозрядні порти введення-виведення, послідовний інтерфейс для програмного завантаження пам'яті, модуль переривань, сторожовий таймер, таймери-лічильники та вбудовані аналогово-цифрові перетворювачі.

Широка номенклатура AVR дає можливість вибрати мікроконтролер з мінімальною апаратною надлишковістю та найменшою вартістю. Серійно випускаються мікроконтролери AVR малої (Tiny), середньої (Classic) та великої (Mega) потужності.

Розглянемо архітектуру найбільш розповсюдженого мікроконтролера AVR Classic (рис.80). До складу його мікропроцесорного ядра CP, входить арифметико-логічний пристрій ALU, регістри загального призначення GnR, лічильник команд, або програмний лічильник PC, вказівник стеку SP і регістр стану PSW. Воно також містить регістр команд RC і дешифратор команд DCC. Мікроконтролер має енергонезалежну постійну пам'ять програм з електричним стиранням FLASH ROM обсягом (1-8) Кбайт та енергонезалежну пам'ять даних EEP ROM, які завантажуються як за допомогою спеціального програматора, так і за допомогою послідовного периферійного інтерфейсу SPI. FLASH – пам'ять програм, доступна для запису, читання і стирання під час роботи мікроконтролера у всьому діапазоні допустимої напруги живлення.

Пам'ять даних мікроконтролера AVR Classic розділена на три частини: енергонезалежна пам'ять даних EEP ROM обсягом (64-512) байт, яка використовується для запису констант, таблиць перекодувань, коефіцієнтів і завантажується подібно пам'яті програм ; оперативна статична пам'ять DATE RAM з лінійною організацією та внутрішня регістрова пам'ять. До

складу регістрової пам'яті входять 32 регістри загального призначення GnR, що об'єднані у регістровий файл, службові регістри портів введення/виведення $PA_{(8)}$, $PB_{(8)}$, $PC_{(8)}$, $PD_{(8)}$ та службові регістри керування.

Обсяг регістрової пам'яті є фіксованим і дорівнює 96 байтам. Під регістри загального призначення виділено 32 молодших байта, а під регістри портів, вказівник стеку, регістр стану та регістри управління – 64 байта.

Мікроконтролер має гарвардську архітектуру, яка забезпечує повний логічний і фізичний розподіл адресних просторів та інформаційних шин для звернення до пам'яті програм і до пам'яті даних (рис.3).

Внаслідок одночасної роботи мікропроцесора з пам'яттю програм і пам'яттю даних, значно зростає продуктивність його роботи. Наступним кроком збільшення швидкодії AVR є використання технології конвейєризації, яка скорочує цикл "вибір-виконання" команди (рис.73).

Регістри загального призначення GnR *мікропроцесорного ядра* створюють регістровий файл швидкого доступу. Кожний з регістрів є однобайтовим і безпосередньо з'єднаний з арифметико-логічним пристроєм ALU. Таким чином, AVR містить 32 "регістри-акумулятори", що разом з конвейєрною обробкою дозволяє виконувати цикл вибору та виконання команди за один машинний такт. Наприклад, два операнди вилучаються з регістрового файлу, виконується команда і результат знову записується в регістровий файл на протязі одного такту.

Шість з 32 регістрів файлу (рис.80) використовуються як три 16-розрядних вказівника адреси при попередній адресації даних. Один з них використовується для доступу до таблиць перекодувань, записаних у постійної пам'яті даних. Використання трьох 16-бітних вказівників (x,y та z Points) підвищує швидкість пересилання даних. Регістровий файл, що займає 32 молодші байти в загальному адресному просторі пам'яті даних, забезпечує швидкий доступ до неї, так зване "швидке контекстне перемикання".

Під час переходів до виконання підпрограм або процедур обробки переривань, поточний стан програмного лічильника PC зберігається у стеці, розміщеному в загальному оперативному пристрої пам'яті даних DATE RAM. 16-розрядний вказівник стеку SP знаходиться в мікропроцесорі CP і є доступним для читання та запису.

Система команд AVR містить 120 фіксованих двобайтних інструкцій, що дозволяє поєднувати в одній команді код операції і операнда. Основні команди мікроконтролера AVR діляться на такі групи:

- команди пересилання даних ;
- команди арифметичних дій ;
- команди логічних операцій ;
- команди роботи з бітами ;
- команди умовного розгалуження ;
- команди безумовного розгалуження.

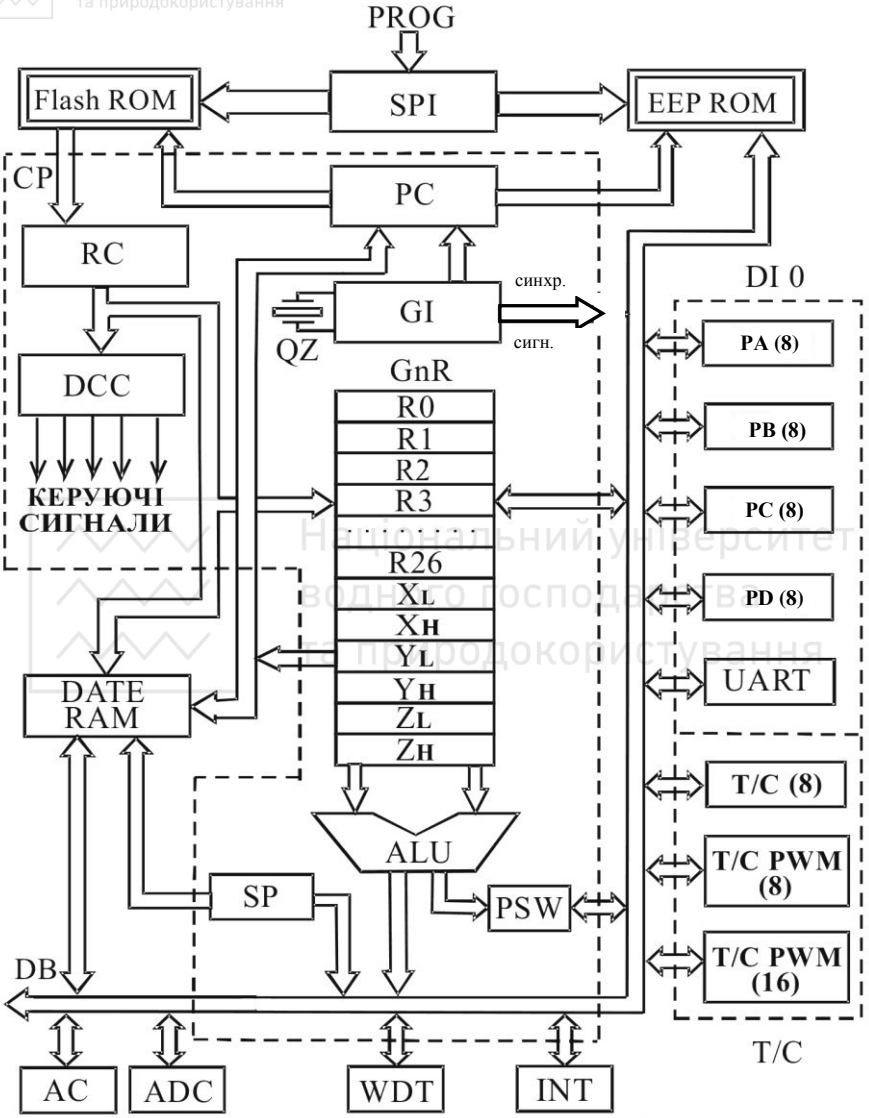


Рис. 80 Структурна схема однокристального мікроконтролера AVR CLASSIC з RISC-архітектурою

В мікроконтролерах AVR використовується однорівневий конвейєр. При зверненні до пам'яті програм, команда виконується за один період тактової частоти: формується і зчитується код операції, операція виконується, фіксується результат і одночасно зчитується код нової операції. Так відбувається конвейєрна обробка інформації. Часозадаючим елементом мікроконтролера є внутрішній генератор тактових імпульсів GI, стабілізований кварцовим резонатором QZ.

8-розрядні порти введення/виведення $PA_{(8)}$, $PB_{(8)}$, $PC_{(8)}$, $PD_{(8)}$, мають 32 незалежних лінії, кожна з яких програмується на введення або виведення інформації. Потужні вихідні підсилювачі (драйвери), забезпечують струмову навантажувальну спроможність на лінію порту $I_н$=20 mA, що дає можливість безпосередньо підключати до виходів мікроконтролера світлодіоди і біполярні транзистори.

До складу мікроконтролера AVR входять також модулі, які розширюють його функціональні можливості. В першу чергу це асинхронний прийомопередавач UART, що створює послідовний інтерфейс введення/виведення, розглянутий вище.

Мікроконтролер AVR містить також таймери /лічильники $T/C_{(8)}$, $T/C\ PWM_{(8)}$; $T/C\ PWM_{(16)}$ (рис.80).Вони мають власні подільники частоти, які програмним способом під'єднуються до внутрішнього генератора тактових імпульсів GI, або до зовнішнього генератора опорної частоти. Якщо на вхід модуля T/C надходять сигнали ззовні, з порту "$PB_{(8)}$", то він працює у режимі лічильника, а якщо імпульси надходять з внутрішнього генератора GI, то модуль переходить в режим таймера, що формує часові затримки. При цьому в модуль T/C попередньо заноситься величина часу затримки і здійснюється його пуск, а мікроконтролер виконує свої функції. Після формування імпульсу заданої тривалості таймер/лічильник подає сигнал на внутрішню шину даних DB. При цьому мікроконтролер здійснює повторний пуск модуля T/C і процес формування імпульсів повторюється. Висока ефективність використання модуля таймер/лічильник T/C забезпечується модулем переривань INT.

Інтерфейс послідовного порту SPI (Serial Periferial Interfase) призначений для послідовного введення даних і забезпечує перезапис робочих програм в мікроконтролері AVR після його встановлення на платі. Для цього масив пам'яті програм FLASH ROM ділиться на блок завантажника (програми, що керує перезаписом FLASH-пам'яті програм) і блок для розміщення робочих програм. Програму-завантажник створює розробник і вона повинна бути попередньо записана у пам'ять спеціальним зовнішнім програматором.

Сторожовий таймер WDT (WATCH DOG) призначений для захисту мікроконтролера від збоїв в роботі. Програма, що працює без збоїв, періодично скидає сторожовий таймер, не допускаючи його переповнення. В

нього вбудований додатковий RC-генератор, що працює на частоті 1МГц. На вході WDT увімкнено подільник вхідної частоти з програмованим коефіцієнтом ділення, що дозволяє регулювати інтервал скидання мікроконтролера.

Аналоговий компаратор AC порівнює напруги сигналів на нульовому та першому входах порту $PB_{(8)}$, а результат порівняння подає на 6-ий вхід порту $PC_{(8)}$.

Аналогово-цифровий перетворювач ADC побудований за класичною схемою послідовних наближень(рис.80).Його розрядність складає 10 бітів, а час перетворення встановлюється програмно – зміною частоти подільника, що входить до складу АЦП, і дорівнює 65÷260 мкс.

Мікроконтролери AVR програмно переводяться в один із режимів обмеженого енергоспоживання: холостого ходу ; мікроспоживання ; зберігання енергії, заглушення шуму, а також в основний або додатковий режими очікування. Слід зазначити, що співвідношення "продуктивність/енергоспоживання/вартість" для мікроконтролерів AVR є найкращими на світовому ринку 8-розрядних мікроконтролерів.

24. Використання однокристальних мікроконтролерів для керування машинами

Машини є складними технічними системами, що містять двигуни, ходове обладнання, приводи вузлів, контрольні прилади та керуючі пристрої. Їх робота характеризується великою кількістю параметрів, а також тим, що зміна збурюючих впливів здійснюється за випадковими законами. Тому управління машинами з використанням бортових комп'ютерів — однокристальних мікроконтролерів дає значний ефект: забезпечує зменшення споживання палива, кількості шкідливих домішок у вихлопних газах, вібрацій та рівнів шумів, підвищення безпеки руху у складних умовах, а також діагностику обладнання і збільшення терміну його роботи.

Високі темпи впровадження мікросхемотехніки і мікроЕОМ для обладнання їми машин підтверджують такі цифри. В 1987 році світовий обсяг всіх виробів промислової електроніки для автоматизації виробництва склав 10 млрд. дол, а в 1990 обсяг мікроелектроніки і мікроЕОМ лише для автомобілів досяг 14,4 млрд. дол. і сьогодні ця тенденція продовжує зростати.

24.1. Комплексна мікропроцесорна система керування бензиновими двигунами

В сучасних автомобілях з бензиновими двигунами передбачаються комплексні мікропроцесорні системи керування силовими агрегатами на базі однокристальних мікроконтролерів.

До складу комплексної мікропроцесорної системи керування бензиновим двигуном БД з розподіленим впорскуванням палива (рис.81) входить паливний бак ПБ, електронасос ЕН, фільтр Ф, паливна магістраль ПМ, гідравлічний регулятор тиску РТ, повітряний фільтр ПФ, повітряна дросельна заслінка ПЗ, свічки запалювання СЗ, розподільник запалювання РЗ та котушка запалювання КЗ. Паливо з паливного баку ПБ подається

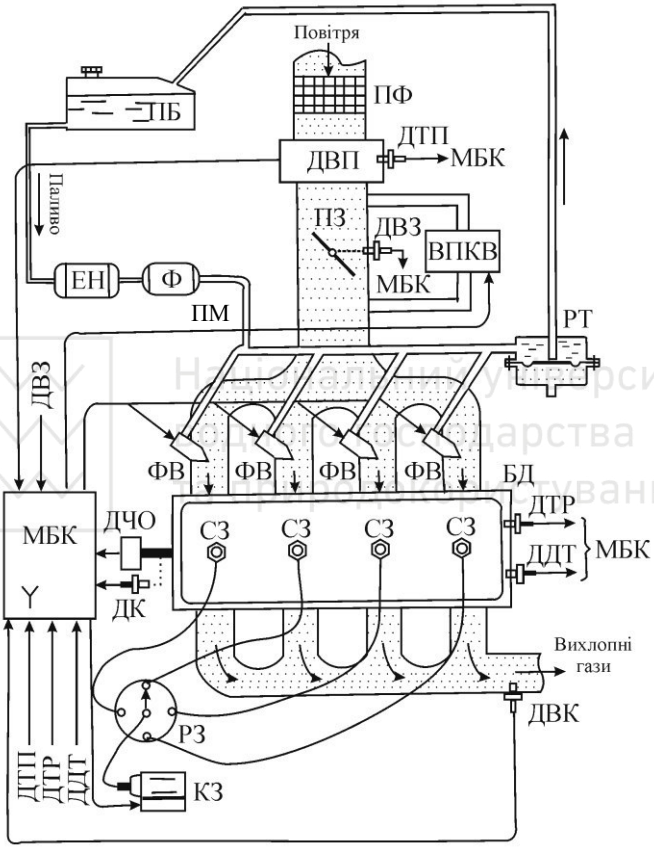


Рис.81. Комплексна мікропроцесорна система керування бензиновим двигуном з розподіленим впорскуванням палива

електронасосом ЕН через фільтр Ф до паливної магістралі ПМ і далі до форсунок впорскування палива ФВ під сталим тиском p=0,25 МПа. В разі підвищення тиску у паливній магістралі ПМ, регулятор тиску прямої дії РТ відкриває свій мембранний клапан і повертає надлишок палива у паливний бак ПБ. Контроль параметрів двигуна здійснюється датчиками витрати

повітря ДВП, температури повітря ДТП, відкривання (кута повороту) дросельної заслінки ДВЗ, температури охолоджувальної рідини ДТР, контролю детонації ДДТ, вмісту кисню у вихлопних газах ДВК, а також частоти обертання колінчатого валу ДЧО та кута його повороту ДК. Сигнали з датчиків надходять на входи 8-розрядного мікропроцесорного блока керування МБК. Зв'язок між сигналами датчиків та керуючими сигналами визначає програма ОМК, в яку трансльований алгоритм керування двигуном. Виконавчими елементами системи є форсунки впорскування палива ФВ, які обладнані спеціальними електромагнітними голчастими клапанами. При протіканні струму через обмотку електромагніту, рухоме осердя клапана втягується в котушку і з'єднаний з осердям голчастий клапан відкривається. Обмотка електромагніту має значну індуктивність, внаслідок чого може виникати затримка закривання клапана після припинення сигналу. Для підвищення швидкодії форсунки, зменшують кількість витків обмотки електромагніту і послідовно з нею вмикають обмежувальний резистор. Разом з цим, виконавчий пристрій керування частотою обертання колінчатого валу на холостому ході ВПКВ (ISCV) (рис.81) регулює витрату повітря за допомогою осьового переміщення клапана гвинтовим механізмом, жорстко з'єднаним з ротором крокового двигуна. Величина переміщення відповідає кількості імпульсів, що надходять з портів виведення ОМК. До виконавчих елементів відноситься також розподільник запалювання РЗ.

Мікропроцесорна комплексна система забезпечує оптимальні умови роботи бензинового двигуна внутрішнього згорання за рахунок керування впорскуванням палива, регулювання кута випередження запалювання та частоти обертання колінчатого валу на холостому ході. Вона також виконує функції діагностики, підвищує чистоту вихлопних газів та економічність двигуна.

Керування впорскуванням палива здійснюється на підставі сигналів датчиків за допомогою розрахунку ОМК часу впорскування, виходячи із оптимального співвідношення палива і повітря в паливній суміші. Завдяки тому, що в паливній магістралі тиск стабілізується гідравлічним регулятором РТ, - об'ємна витрата палива через форсунки є також сталою величиною :

$$Q_{пл} = \mu_c \omega_c \sqrt{\frac{2p}{\rho_{пл}}} = \text{const},$$

де μ_C- коефіцієнт витрати сопла форсунки, віднесений до площі його перерізу ω_C; p=const-тиск на вході у форсунку, МПа; $\rho_{пл}$-густина палива, кг/м^3.

Тому кількість впорскуваного палива визначається тільки часом відкривання електромагнітного клапана форсунки "t"$_в$:

$$W_{пл}=Q_{пл} * t_в.$$

При усталеному русі машини, впорскування палива здійснюється синхронно: на один оберт колінчатого валу виконується одне впорскування. Але під час його прискорення, для підвищення потужності двигуна, використовується асинхронне впорскування з врахуванням зміни умов роботи двигуна. Час синхронного впорскування визначається за формулою :

$$T_{св}=k*t_{ов}+\Delta t_u ,$$

де k-коефіцієнт корекції; $t_{ОВ}$-основний(базовий) час впорскування, с ; Δt_U-поправка часу впорскування на зміну напруги живлення, с .

За основний час впорскування в циліндр надходить кількість палива, яка необхідна для створення оптимального співвідношення паливо/повітря. При цьому маса повітря, що надходить в циліндр двигуна за один цикл, розраховується по даним датчиків *витрати повітря і частоти обертання колінчатого валу двигуна*. Час спрацювання електромагнітного клапана форсунки змінюється при змінах напруги бортової мережі "U", і тому вводиться поправка часу Δt_U. Разом з цим, під час прогрівання двигуна, виконуються корекції кількості впорскуваного палива на підставі сигналів *датчика температури охолоджувальної рідини та датчика температури повітря*. Обидва датчики виконані на базі напівпровідникових терморезисторів з від'ємним температурним коефіцієнтом – термісторів, активний опір яких суттєво зменшується при нагріванні і зростає при охолодженні. Датчик температури повітря встановлюється безпосередньо у витратомірі повітря, а датчик температури охолоджувальної рідини, – як правило, біля термостату.

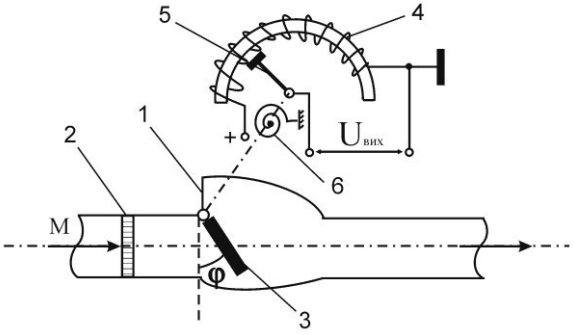


Рис.82. Датчик витрати повітря

Датчик витрати повітря містить витратомір повітря з потенціометричним перетворювачем (рис.82). Він складається з повітрозбирача 1, фільтра 2, рухомої заслінки 3, потенціометра 4 з повзунком 5 та тарованої пружини 6. Повітря надходить у двигун через повітряний фільтр 2 і повертає заслінку 3 на кут φ. При цьому сила

швидкісного напору повітря зрівноважується силою пружності тарованої пружини 6. Таким чином, масова витрата повітря М перетворюється на кут повороту заслінки φ, переміщення повзунка потенціометра 5 та зміну вихідної напруги $U_{ВИХ}$, яка змінюється прямопропорційно витраті $U_{ВИХ}=f(M)$.

В звя'зку *з введенням вимог на склад вихлопних газів* однокристальні мікроконтролери ОМК (бортові мікроЕОМ) забезпечують їх одночасне очищення по компонентам СО, НС та NO_X . Для цього у випускній системі бензинового двигуна встановлюється датчик вмісту кисню O_2 у вихлопних газах (ДВК) і створюється канал зворотного зв'язку для автоматичної стабілізації стехіометричного складу паливної суміші. Він перетворює концентрацію кисню O_2 у вихлопних газах "λ" на пропорційну величину електрорушійної сили (е.р.с.) "Е". Конструкція датчика наведена на рис.83а, а його статична характеристика E=f(λ),- на рис.83б.

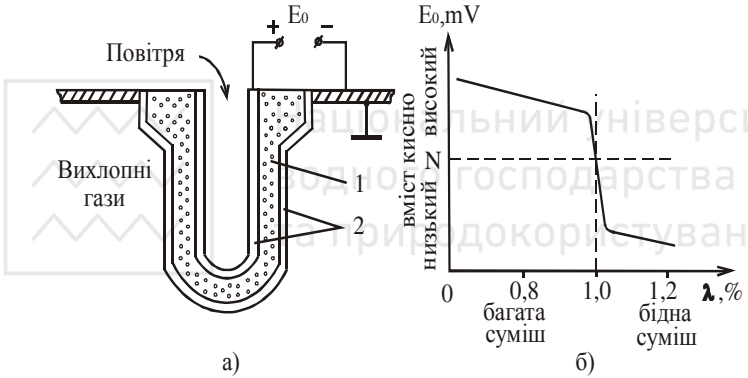


Рис.81. Датчик вмісту кисню : а) конструкція ; б) статична характеристика .

Він видає сигнали про відхилення від стехіометричного складу паливної суміші, яка надходить в циліндри двигуна БД (рис.81). Датчик виготовляється з порошку двоокису цирконію у вигляді пробірки 1, зовнішня та внутрішня стінки якої покриті пористою платиною 2. На зовнішню поверхню датчика діють вихлопні гази, а на внутрішню-атмосферне повітря. Е.р.с., що виникає між його зовнішньою та внутрішньою поверхнями, має однозначну залежність від концентрації кисню O_2 у вихлопних газах. Бортовий однокристальний мікроконтролер на підставі сигналів перетворювача ДВК забезпечує оптимальний склад паливної суміші.

Іншою важливою функцією мікропроцесорної системи (рис.81) є керування кутом випередження запалювання θ. Його величина визначається як сума базового кута випередження запалювання θ_0 та поправки $\Delta\theta$:

$$\theta=\theta_0\pm\Delta\theta.$$

При холодному двигуні має місце раннє запалювання ($\Delta\theta>0$), а при високих температурах – більш пізнє ($\Delta\theta<0$). Під час руху машини, залежність оптимального кута випередження запалювання від частоти обертання колінчатого валу двигуна та кількості споживаного повітря має такий вигляд (рис.84):

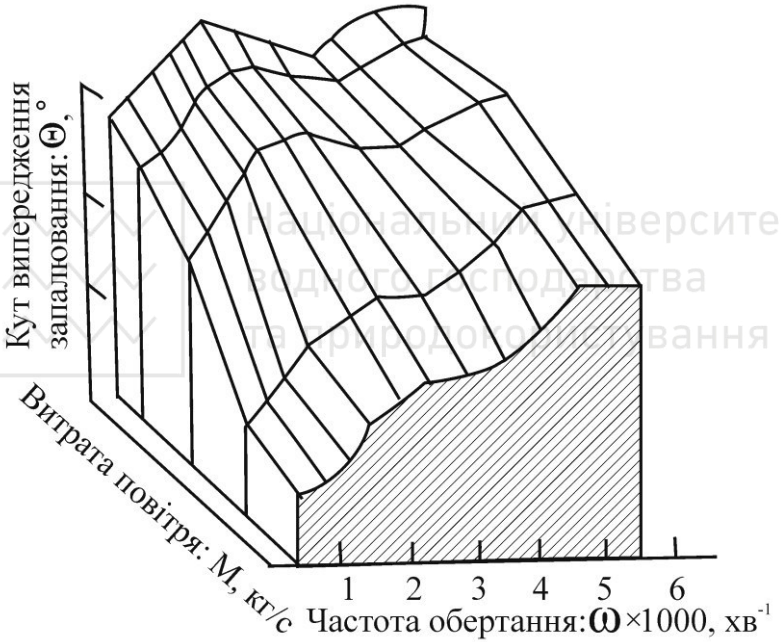


Рис.84. Зміна величини кута випередження запалювання під час руху машини

Щоб забезпечити запалювання в розрахунковий момент часу, в комутатор запалювання SW_3 подаються комутуючі імпульси, передній і задній фронти яких визначають початок і кінець протікання струму через котушку запалювання КЗ. В системі керування запалюванням використовують датчики кута повороту колінчатого вала ДК, витрати повітря ДВП, температури охолоджувальної рідини ДТР та відкривання (кута повороту)

дросельної заслінки ДВЗ. По сигналу запалювання з МБК потужний вихідний транзистор VT_K, що увімкнений на виході комутатора запалювання SW_3 і працює у ключовому режимі, короткочасно відкривається, а потім закривається. Вимкнення струму, який протікає у первинній обмотці котушки запалювання КЗ, приводить до виникнення високої напруги у її вторинній обмотці, яка через розподільник Р3 надходить на свічки запалювання С31÷С34 (рис. 85).

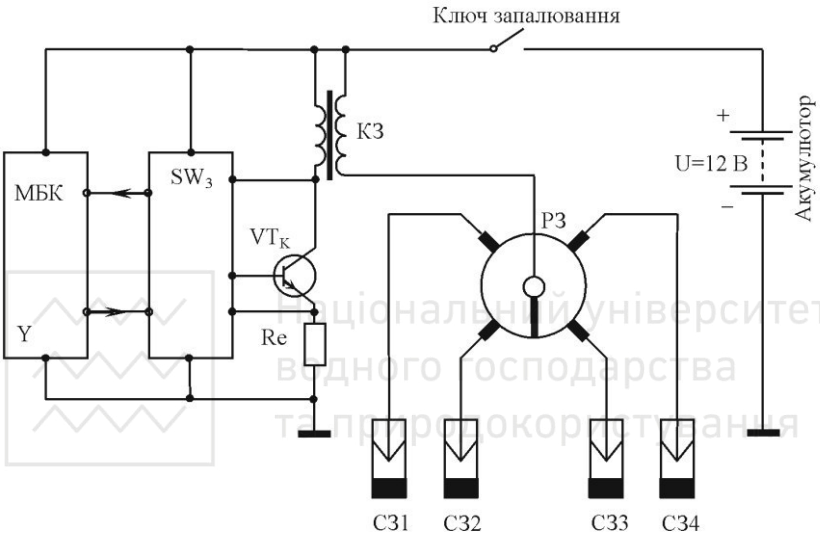


Рис.85. Структурна схема системи запалювання:
МБК – мікропроцесорний блок керування двигуном; SW_3 - комутатор запалювання; VT_K – вихідний комутуючий транзистор; Re – емітерний опір; КЗ – котушка запалювання; Р3 – розподільник запалювання; С31÷С34 – свічки запалювання.

Керуючий сигнал задає час відкриття транзистора VT_K і узгоджує роботу системи запалювання з роботою системи впорскування палива.
Значний ефект надає визначення кута випередження запалювання, близького до граничного, за яким виникає детонація. При цьому підвищується к.к.д. двигуна, його потужність і економічність, а також стає можливим використання палива з різними октановими числами. Керування здійснюється на підставі сигналів датчика детонації ДДТ, встановленого на блоці циліндрів двигуна БД (рис.81.). Датчик детонації містить пьєзоелемент,

вбудований в жорсткий корпус. При виникненні детонації частота коливань блока циліндрів співпадає з власною частотою коливань дископодібного пьезоелемента, на гранях якого при резонансі з'являється змінна напруга з дуже високою амплітудою.

Розпізнавання детонації робиться порівнянням поточного сигналу датчика з середньою величиною його амплітуди при відсутності детонації на протязі певного часу після надходження сигналу запалювання. Її оцінка визначається шляхом підрахунку кількості амплітуд в сигналі датчика, величина яких більша стандартного значення, яке є характерним для початку детонації (рис. 86). В МБК після їх обробки формуються сигнали на зміну кута випередження запалювання до оптимального значення (зменшення при наявності детонації або збільшення при її відсутності).

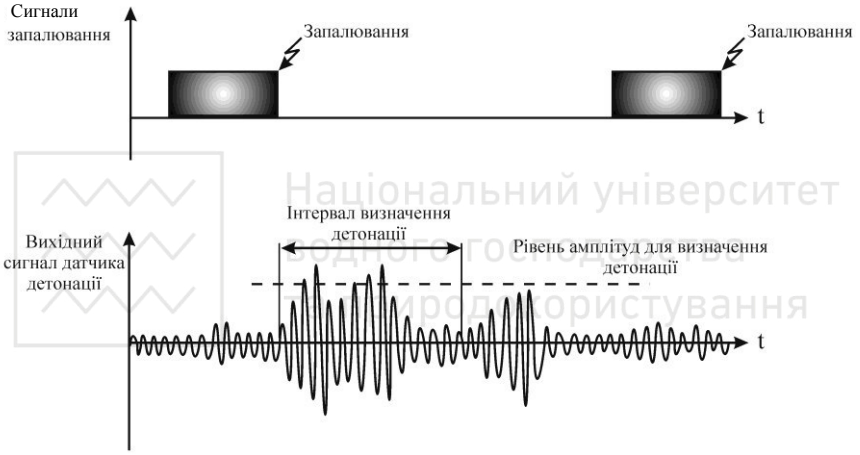


Рис.86. Часова діаграма процесу детонації

Комплексна мікропроцесорна система керування бензиновим двигуном забезпечує також оптимальну частоту обертання колінчатого валу двигуна на холостому ході. При цьому зменшується витрата палива за рахунок зміни частоти обертання при коливаннях навантаження двигуна. Автоматичне керування забезпечує оптимальну частоту обертання, виходячи з сигналів датчиків частоти обертання колінчатого валу ДЧО, кута повороту повітряної дросельної заслінки ДВЗ, температури охолоджувальної рідини ДТР (рис. 81), а також сигналу наявності холостого ходу, що надходить з мікропроцесорного блоку керування трансмісією. За допомогою клапана керування частотою обертання на холостому ході ISCV регулюється кількість повітря, що проходить по обводному каналу повз дросельну заслінку. Як привод клапана використовується крокoвий електродвигун (рис. 87).

Пускові характеристики двигуна покращуються, якщо надходження повітря через обвідний канал є максимальним. Після пуску двигуна витрата повітря $M_П$ через обвідний канал регулюється ISCV в залежності від температури охолоджувальної рідини і зменшується при її зростанні (до 60°С). Далі починається режим прогрівання і степінь відкривання каналу продовжує регулюватись до того моменту , коли частота обертання колінчатого валу стане оптимальною. Її величина розраховується мікропроцесорним блоком керування по навантаженню двигуна і регулюється за принципом відхилення, який полягає у зменшенні різниці між розрахунковою і поточною частотами обертання колінчатого валу.

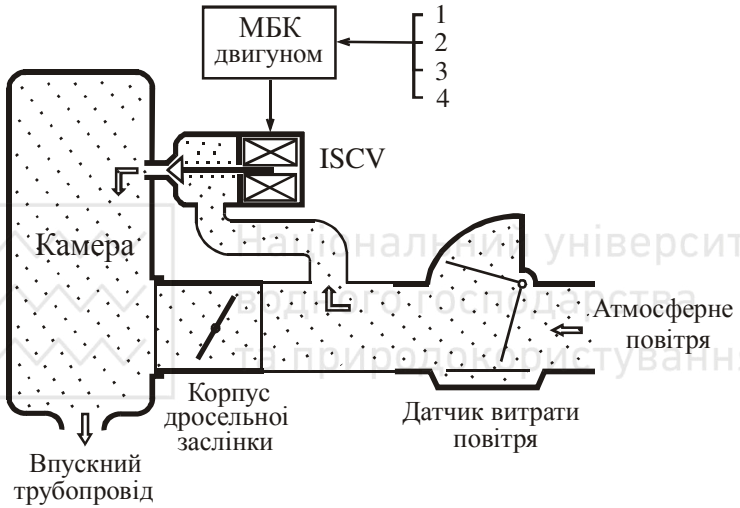


Рис.87 . Система керування частотою обертання колінчатого вала на холостому ході : ISCV - клапан керування частотою обертання на холостому ході ; 1 - сигнал частоти обертання колінчатого вала; 2 - сигнал наявності холостого ходу(з МБК трансмісії); 3 - сигнал температури охолоджувальної рідини; 4 - сигнал кута повороту дросельної заслінки.

Дуже важливою функцією мікропроцесорної системи керування двигуном є діагностика – виявлення, запам’ятовування та відображення на дисплеї або табло відхилень від норми, що виникають в аварійних елементах управління двигуном і трансмісією (в датчиках, виконавчих пристроях, бортовій мережі та безпосередньо в обладнанні двигуна, трансмісії і т. д.). Якщо величина сигналу, що надходить в МБК від будь-якого датчика, або сукупність сигналів від кількох датчиків, суттєво відрізняється від можливих значень, то однокристальний мікроконтролер ОМК МБК аналізує ситуацію.

В результаті визначаються дефекти і ці дані записуються в оперативну пам'ять, або в енергонезалежну пам'ять для наступного виведення на дисплей станції технічного обслуговування. Одночасно результати контролю роботоспроможності елементів системи у вигляді діагностичного коду надходять на дисплей та інформують водія або оператора. Як діагностичний код в більшості систем використовують двохрозрядний десятковий код. Після усунення пошкодження його діагностичний код стирається і з'являється код роботоспроможного стану.

Керуючим пристроєм є мікропроцесорний блок керування МБК, структурна схема якого наведена на рис. 88. Основним модулем МБК є однокристальний 8-розрядний мікроконтролер ОМК з вбудованим аналогово-цифровим перетворювачем АЦП, схема попередньої обробки дискретних сигналів СПО, схема перетворення сигналів датчика детонації СПД, схема вихідної обробки сигналів мікроконтролера СВО, та стабілізований блок живлення (U=+5В).

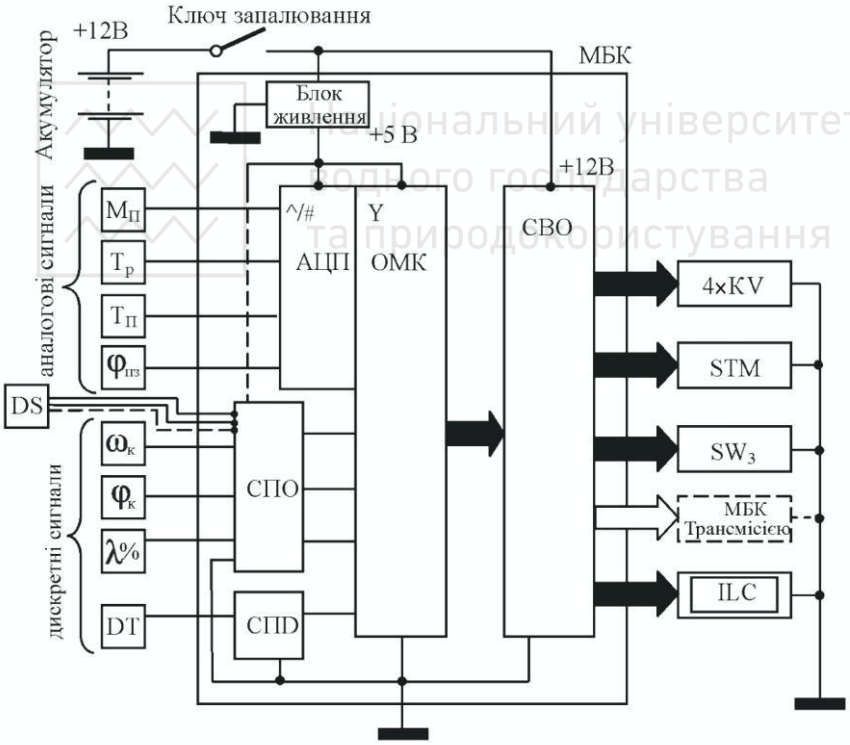


Рис.88. Структурна схема мікропроцесорного блока керування *МБК бензиновим двигуном з розподіленим впорскуванням палива*

Аналогові сигнали витрати повітря $M_П$, температури охолоджувальної рідини $T_Р$, температури повітря $T_П$ та кута відкривання дросельної заслінки $\varphi_{пз}$ надходять на входи АЦП і перетворюються на цифровий двійковий код. Дискретні сигнали частоти обертання колінчатого валу $\omega_к$, кута його повороту $\varphi_к$ та вмісту кисню у вихлопних газах λ% надходять на входи схеми попередньої обробки дискретних сигналів. Мікроконтролер отримує напругу живлення +5В від блоку стабілізованого живлення, що входить до складу МБК. Слід зазначити, що сигнали від дискретних датчиків та перемикачів можуть перевищувати допустиму напругу (+5В) (1); мати змінну полярність (2); містити завади і шуми (3) або кидки напруги (4). Ці сигнали, пройшовши схему попередньої обробки СПО, перетворюється до нормованого вигляду і вводяться в порти мікроконтролера (рис. 89).

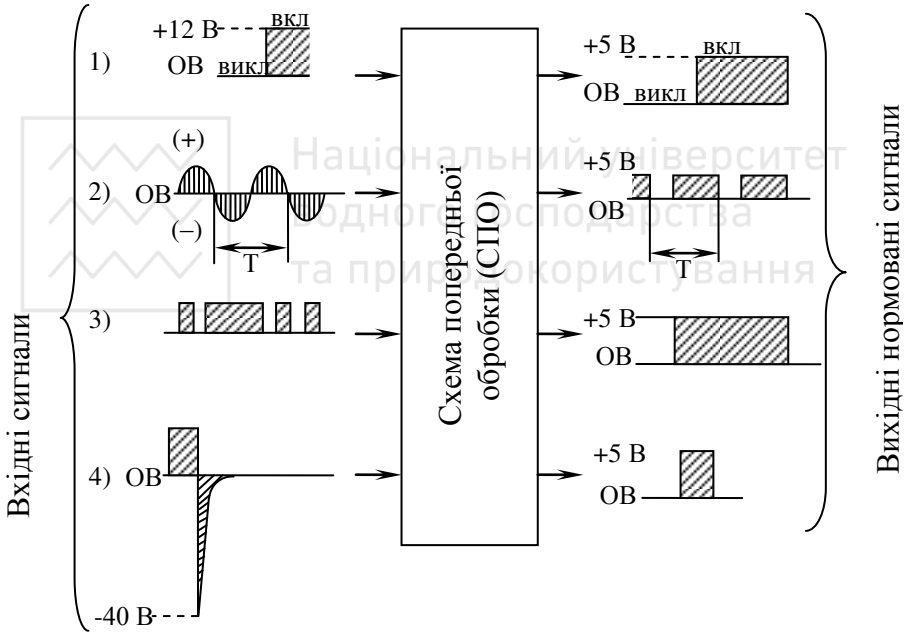


Рис. 89. Попередня обробка дискретних вхідних сигналів

Більшість виконавчих елементів і пристроїв комплексної мікропроцесорної системи керування двигуном використовують напругу акумуляторної батареї +12 В і тому керуючі сигнали ОМК, що живиться від напруги +5 В, попередньо надходять на узгоджувальну схему вихідної обробки СВО. До її виходів під'єднуються обмотки електромагнітів 4-х

форсунок впорскування палива (4×KV), обмотка крокового електродвигуна системи ISCV (STM) та комутатор запалювання (SW_3).

На входи СПО надходять також діагностичні сигнали DS, які після аналізу ОМК через СВО виводяться на індикатор рідких кристалів ILC. Окремий вихідний канал створює датчик детонації DT, сигнали з якого надходять на ОМК через схему перетворення CPD. Окремі сигнали з виходів СВО проходять на мікропроцесорний блок керування МБК трансмісією.

Система з розподіленим впорскуванням є найбільш досконалою, але дорожча в порівнянні із системою мікропроцесорного керування карбюратором, яка застосовується на автомобілях з двигунами малого об'єму. В цьому випадку електромагнітний клапан керує перерізом жиклера. При протіканні струму через обмотку електромагніта клапан закриває отвір жиклера, а при вимкненні струму – відкриває його. Таким чином, по сигналам МБК регулюється співвідношення паливо/повітря. Слід зазначити, що показники якості регулювання цим способом значно нижчі в порівнянні з розподіленим впорскуванням.

Компромісною з точки зору якості регулювання і собівартістю є система центрального впорскування. Процес керування центральним впорскуванням є аналогічним керуванню з розподіленим впорскуванням. Відміна полягає в тому, що в системі центрального впорскування в дросельній камері як спільна конструкція розміщені форсунка впорскування, регулятор тиску, датчик відкривання повітряної дросельної заслінки та дросельна заслінка. Єдина форсунка розташована над дросельною заслінкою і впорскуване паливо самотужки розподіляється по циліндрам. При цьому необхідно, щоб одна форсунка забезпечувала роботу двигуна у всьому діапазоні його режимів. Цей спосіб забезпечує більш високу якість керування подачею палива порівняно з карбюратором, але нижчу порівняно з розподіленим впорскуванням.

24.2. Комплексна мікропроцесорна система керування дизельним двигуном

Машини, що призначені для будівельних та гідромеліоративних робіт, мають переважно дизельні двигуни. Мікропроцесорне керування дизельними двигунами оптимізує режими їх роботи. Мікропроцесорний блок керування МБК обробляє інформацію про процеси у двигуні, яка надходить з датчиків, і видає керуючі сигнали, що визначають момент впорскування та оптимальну кількість палива, паливному насосу.

До складу двигуна входять блок циліндрів, форсунки впорскування палива, паливний насос високого тиску, повітрязабірна система та система турбонаддуву. Інформація про стан двигуна надходить з таких датчиків:

- кута повороту колінчатого валу $\varphi_к$;
- температури повітря, що надходить у двигун $T_п$;
- кутів відкривання дросельних повітряних заслінок $\varphi_{пз}$;
- тиску повітря, що надходить у двигун $P_п$;
- запалювання (спалаху) паливної суміші $L_{пс}$;
- температури охолоджувальної рідини $T_р$;
- частоти обертання колінчатого валу $\omega_к$.

Робота паливного насоса високого тиску здійснюється за допомогою кількох електромагнітних клапанів, які керують:

- моментом початку впорскування палива;
- процесом впорскування палива, закриваючись та відкриваючись за сигналами, що надходять з МБК;
- припиненням подачі палива.

Керування дросельними повітряними заслінками здійснюється виконавчим пристроєм, основними елементами якого є крокові електродвигуни з гвинтовими механізмами та пневматичний сервомеханізм.

Структурна схема МБК дизельним двигуном є подібною до схеми МБК бензиновим двигуном (рис 88). Його основним модулем є 8-розрядний однокристальний мікроконтролер ОМК з вбудованим АЦП. На аналогові входи ОМК надходять сигнали з датчиків: температури повітря $T_п$, тиску повітря $P_п$, температури охолоджувальної рідини $T_р$ та кута відкривання дросельних повітряних заслінок $\varphi_{пз}$. Дискретні сигнали на схему попередньої обробки СПО МБК надходять з таких датчиків: кута повороту колінчатого валу $\varphi_к$, запалювання паливної суміші $L_{пс}$ та частоти обертання колінчатого валу $\omega_к$. На неї надходять також діагностичні сигнали DS.

Однокристальний мікроконтролер на підставі сигналів про частоту обертання колінчатого валу $\omega_к$ та кутового положення повітряних заслінок $\varphi_{пз}$ попередньо розраховує основну кількість впорскуваного палива. Потім отримана величина коригується з врахуванням сигналів датчиків температури $T_п$ і тиску $P_п$ повітря, що надходить у двигун, а також температури охолоджувальної рідини $T_р$. Таким чином, визначається оптимальна кількість впорскуваного палива. Його оптимальна витрата розраховується, виходячи з тривалості впорскування. Для цього ОМК розраховує момент початку впорскування, який порівнюється із сигналами датчика запалювання в камері згоряння. Цей датчик містить світловод з кварцового скла, вмонтований у стінку циліндра, стійкий до високих температур, та фототранзистор, який перетворює світло на електричний сигнал. ОМК забезпечує збіг реального моменту часу запалювання з його розрахунковим значенням. Таким чином, забезпечується оптимальне споживання палива.

Після обробки інформації і формування керуючих сигналів вони надходять на схему вихідної обробки СВО. Форми вихідних сигналів мають вигляд сигналів рівня , імпульсів певної тривалості із заданою періодичністю, або послідовності певної кількості імпульсів з постійним періодом та змінною фазою (для подачі команд на крокові двигуни). Вихідні сигнали надходять на бази транзисторних ключів, у колекторні кола яких увімкнені обмотки електромагнітних клапанів або крокових двигунів.

Крім оптимізації витрати палива мікропроцесорна система керування двигуном зменшує вібрації на холостому ході та усуває вібрації при його зупинці за допомогою зміни витрати повітря заслінками. Їх виконавчим пристроєм є крокові двигуни, які керують пневматичним сервомеханізмом, що повертає заслінки на кут, пропорційний кількості керуючих імпульсів, які надходять зі схеми вихідної обробки СВО.

При відмовах системи керування повітряні заслінки відкриваються наполовину. На холостому ході система забезпечує оптимальну частоту обертання колінчатого валу $\omega_к=\omega_{кн}$ у різних умовах, що також забезпечує економію палива.

Під час пуску дизеля залежно від температури охолоджувальної рідини $T_р$ та повітря $T_п$ МБК відповідно змінює струм свічки розжарювання, що скорочує час пуску, стабілізує частоту обертання колінчатого валу та зменшує задимленість.

Мікропроцесорна система керування забезпечує діагностику обладнання, тобто визначає, запам'ятовує та відображає на індикаторі рідких кристалів або дисплеї відхилення від норми, які виникають в елементах і вузлах керування двигуном (датчиках, виконавчих механізмах та електропроводці) під час руху машини. Якщо величина сигналу датчика або групи датчиків відрізняються від допустимих значень, ОМК аналізує ситуацію. При цьому запам'ятовуються дефекти елементів, а також обриви та короткі замикання в електропроводці бортової мережі. Ці дані фіксуються в пам'яті і одночасно забезпечується їх індикація.

24.3. Мікропроцесорна система керування трансмісією

Задача мікропроцесорного керування трансмісією полягає у передачі потужності двигуна на ведучі колеса з врахуванням умов руху машини таким чином, щоб одночасно зменшити споживання палива і забезпечити оптимальні тягово-швидкісні характеристики машини.

Для цього ОМК мікропроцесорного блоку керування трансмісією (рис. 90) на підставі сигналів датчиків швидкості машини $\omega_{т,}$ $V_М$, кута відкривання повітряної дросельної заслінки $\varphi_п$, температури охолоджувальної рідини $T_Р$ та сигналів керуючих перемикачів $HSW_Т$, $HSW_{ПТ}$ розраховує моменти перемикання передач і вмикання зчеплення. Одночасно він забезпечує

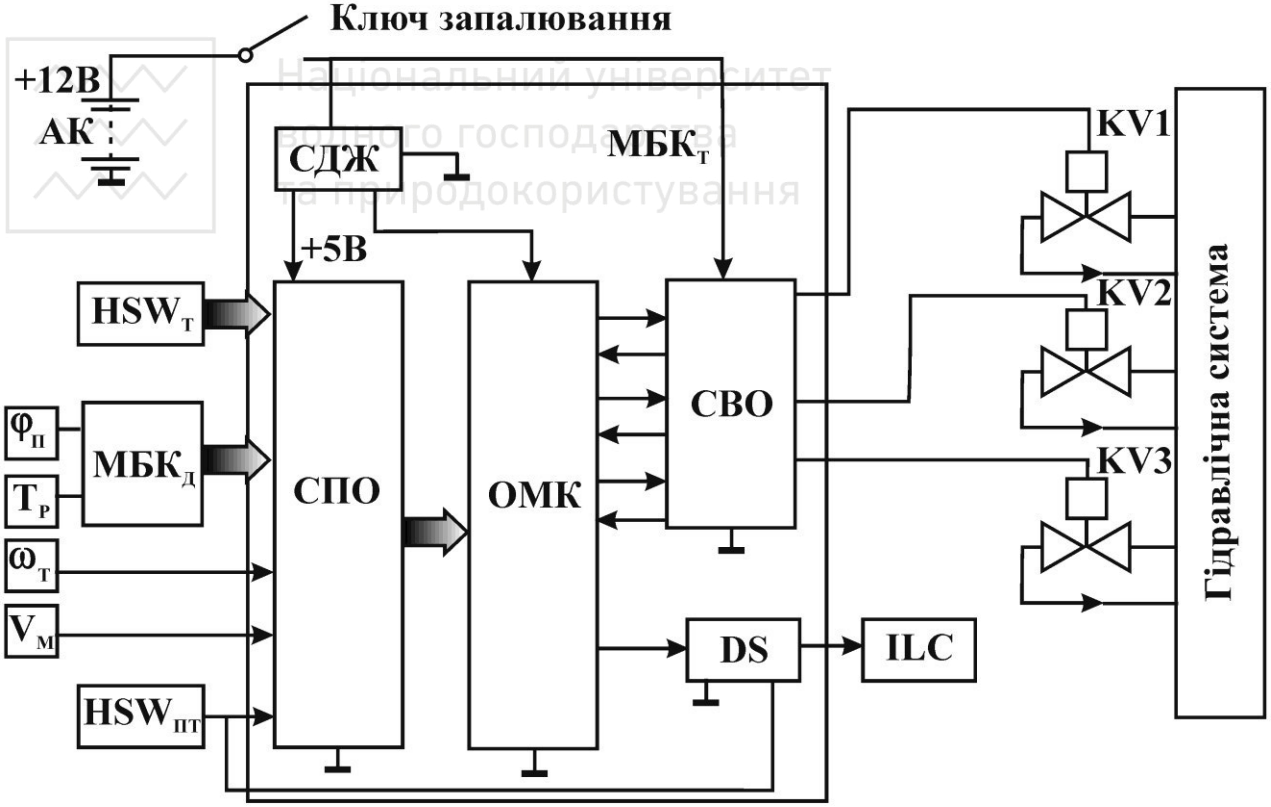


Рис. 90. Структурна схема мікропроцесорного блока керування МБК трансмісією

формування керуючих сигналів на перемикання електромагнітних клапанів гідравлічної системи KV1, KV2, KV3. Перемикач HSW_T визначає режими роботи трансмісії, а $HSW_{ПТ}$ вмикає прискорюючу передачу.

В системі використовується датчик кута відкривання повітряної дросельної заслінки $\varphi_п$ і температури охолоджувальної рідини $T_р$, датчик кутової швидкості $\omega_т$, встановлений в трансмісії, який генерує один імпульс за один оберт вихідного валу та датчик швидкості машини $V_м$, змонтований в спидометрі. ОМК МБК розраховує момент перемикання передач і вмикання зчеплення з врахуванням інерційності перехідних процесів, що виникають при перемиканнях, і видає керуючі сигнали на електромагнітні клапани.

Керування зчепленням здійснюється за допомогою електромагнітного клапана KV1, який забезпечує його вмикання. Для перемикання передач використовуються два електромагнітних клапани KV2, KV3. Сполученням станів „Відкрито - Закрито" цих двох клапанів система задає 4 передачі (першу, другу, третю та прискорюючу). Під час перемикання передач зчеплення вимикається, щоб запобігти різкій зміні навантажувального моменту. Закони керування перемиканням передач в автоматичній трансмісії забезпечують оптимальну передачу енергії двигуна колесам автомобіля з врахуванням необхідних тягово-швидкісних характеристик та економії палива. При цьому, програми оптимізації тягово-швидкісних характеристик та мінімальної витрати палива значно відрізняються, і їх одночасне використання не завжди можливо. Тому в залежності від умов руху за допомогою перемикача HSW_T вибирається режим „ЕКОНОМІЯ" (для зменшення витрати палива), „ПОТУЖНІСТЬ" (для покращення швидкісно-тягових характеристик) або „РУЧНЕ" для перемикання передач водієм. Під час початку руху автоматично вмикається перша передача, завдяки чому скасовується різка зміна навантаження.

На підставі вихідних та вхідних сигналів МБК виявляє відкази електромагнітних клапанів і видає діагностичні сигнали на рідкокристалічний індикатор ILC.

24.4. Мікропроцесорна система керування гальмуванням машини

Значну актуальність має проблема безпеки руху машин в складних дорожніх умовах. При опадах або ожеледиці, в момент блокування задніх коліс машини виникає бічне ковзання, а при блокуванні передніх коліс-автомобіль стає некерованим. Антиблокуюча мікропроцесорна система забезпечує швидку зупинку та підтримує оптимальне зчеплення шин з дорожнім покриттям.

Графічне зображення залежності повздовжнього та бічного зчеплення від ковзання має наступний вигляд (рис.91):

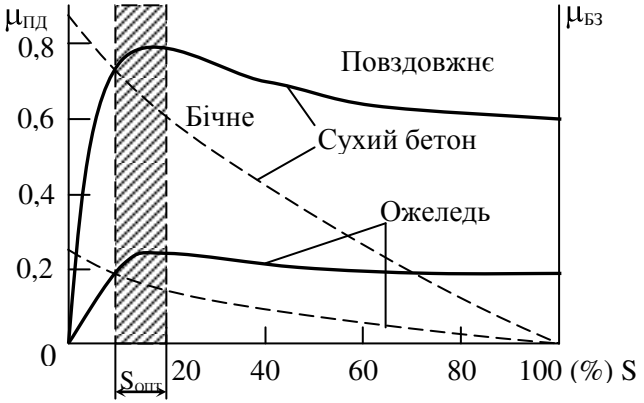


Рис.91. Залежність коефіцієнтів повздовжнього $\mu_{пд}$ та бічного зчеплення $\mu_{бз}$ від ковзання S ($S_{ОПТ}$ - оптимальне ковзання).

З графіку слідує, що при стовідсотковому ковзанні коефіцієнт бічного зчеплення прямує до нуля, а коефіцієнт повздовжнього зчеплення не набуває максимального значення. Забезпечити одночасно оптимальні значення коефіцієнтів повздовжнього і бічного зчеплення неможливо. Разом з тим, якщо виконувати гальмування таким чином, щоб коефіцієнт повздовжнього ковзання знаходився в області максимального коефіцієнта тертя, то можна отримати оптимальне значення сили опору бічного ковзання і забезпечити ефективне гальмування.

Тому управління при ковзанні виконується так, щоб коефіцієнт повздовжнього ковзання у випадку різкого гальмування був рівним 15-20%. Мікропроцесорне керування забезпечує відповідність кутової швидкості коліс ω коефіцієнту ковзання S для кожного з можливих станів дорожнього покриття. Наявність блокування коліс фіксується по різниці розрахункової швидкості кузова і поточної лінійної швидкості коліс. Швидкість кузова розраховується перед початком гальмування, виходячи з кутової швидкості коліс. При цьому лінійна швидкість коліс, при якій починається процес управління $V_{упр}$, приймається меншою на величину $\Delta V_{кл\ 1}$, ніж розрахункова швидкість кузова $V_{кз}$. Швидкість, при якій відбувається ковзання, приймається меншою за швидкість кузова на величину $\Delta V_{кл\ 2}$.

На початку гальмування тиск в гальмівних циліндрах коліс машини підвищується пропорційно тиску на гальма і лінійна швидкість коліс $V_{кл}$ зменшується. Якщо її величина стає меншою за швидкість управління:

$V_{кл} \le V_{упр}$, а від'ємне прискорення за модулем більшим допустимого $|a|>a_{доп}$, то з мікропроцесорного блоку гальмування надходить сигнал на зниження гальмівного тиску $P_Г$, внаслідок чого ковзання припиняється.

Якщо ж і при цьому швидкість коліс продовжує знижуватись і стає меншою швидкості ковзання $V_{кл} \le V_s$, то формується сигнал про надмірно швидке зниження тиску (в циліндрах передніх коліс). Швидкість коліс відновлюється і наближується до нової швидкості кузова. При цьому тиск починає повільно зростати, а при подальшому зростанні прискорення з мікропроцесорного блоку керування надходить сигнал на його більш інтенсивне підвищення. Таким чином, управління тиском гальмівної рідини в гідросистемі з метою уникнення ковзання здійснюється в результаті обробки сигналів про кутову швидкість коліс.

До складу антиблокуючої системи входять: датчики кутової швидкості, мікропроцесорний блок керування і гідравлічний виконавчий механізм з основним та допоміжним електромагнітними клапанами, які безпосередньо керують тиском у гальмівній системі.

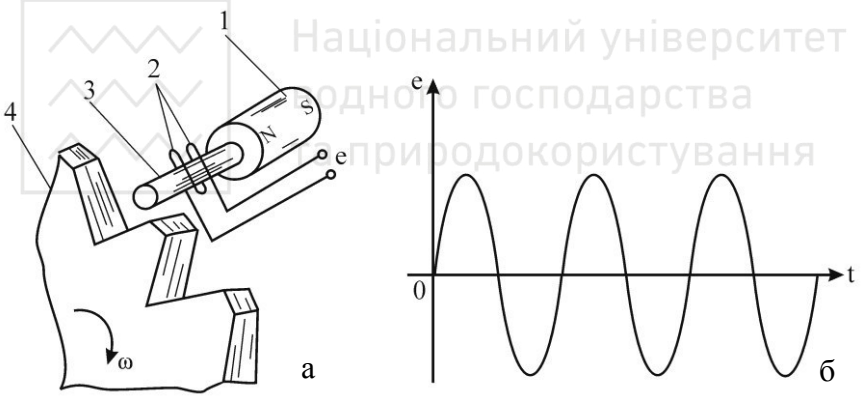


Рис.92. Датчик кутової швидкості коліс машини: а) будова; б) часова діаграма

Датчик кутової швидкості коліс машини (рис. 92а) складається з постійного магніту 1, обмотки *2,* осердя с магнітном'якого матеріалу 3 та зубчастого колеса 4, яке встановлюється на маточині і обертається синхронно із колесом машини. При обертанні зубчастого колеса 4, повітряний проміжок між ним і осердям 3 змінюється за синусоїдним законом. Внаслідок цього змінюється величина магнітного потоку і в обмотці *2* індукується синусоїдна е.р.с., частота якої пропорційна швидкості обертання зубчастого колеса 4, тобто кутової швидкості колеса машини (рис. 92б).

Синусоїдна е.р.с. подається на СПО (рис 89.2), вихідні сигнали з якої надходять в мікропроцесорний блок керування, що виробляє керуючі дії і управляє основним і допоміжним електромагнітними клапанами гальмівної системи.

Обчислення швидкості коліс ґрунтується на підрахунку кількості імпульсів, що надійшли з датчика (рис. 92а) на протязі певного відрізку часу. Існує декілька методів підрахунку вихідних імпульсів. Основний метод ґрунтується на підрахунку кількості імпульсів *n* за визначений проміжок часу T_c. При цьому точність обчислення залежить від швидкості коліс, і при низькій швидкості точність зменшується. Більш досконалим є спосіб обчислення швидкості, заснований на вимірюванні часу T до надходження першого імпульсу після підрахунку кількості імпульсів n за заданий інтервал часу T_a (рис.93).

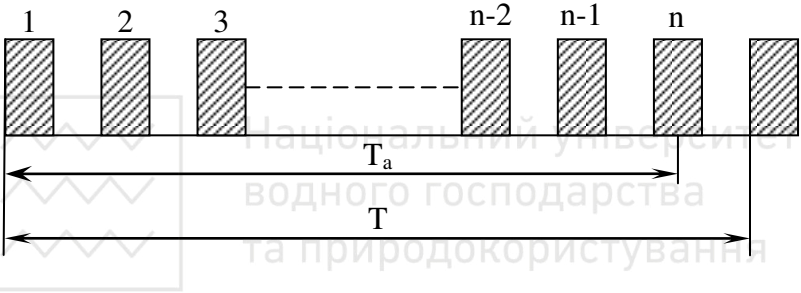


Рис. 93. Спосіб обчислення швидкості з усередненням

При цьому, лінійна швидкість коліс машини дорівнює $V_M = K\ \frac{n}{T}$, де

K - сталий коефіцієнт вимірювань.

Висновок

Будова та експлуатація сучасних машин неможливі без використання мікропроцесорної техніки. Тому в навчальному посібнику розглянуто її основні положення, без яких неможливо зрозуміти комп'ютеризацію сучасних машин при їх експлуатації, ремонті та діагностиці несправностей. Слід зазначити, що машини, які працюють в складних умовах, мають високу експлуатаційну надійність при використанні навіть відносно простих мікропроцесорних засобів.

В навчальному посібнику розглянуто мікропроцесорне керування бензиновим і дизельним двигунами, а також трансмісією та рухомою частиною при гальмуванні в екстремальних умовах. Реалізація цих функцій покращує економічні показники машини та її тягово-швидкісні характеристики, підвищує стійкість та керованість під час руху. Значний розвиток має комп'ютеризація салонів, яка не розглядалась у зв'язку зі стислим викладанням матеріалу мікропроцесорного керування машинами. Об'єднання у єдину мережу всіх мікропроцесорних блоків керування МБК (двигуном, трансмісією, рухомою частиною, салоном і т. п.) надає можливість загального використання інформації, взаємної діагностики, а також виконання функцій МБК, що вийшли з ладу, іншими МБК. В сучасних мікропроцесорних системах керування машинами використовуються їх динамічні моделі, які враховують всі основні параметри: витрату палива, умови руху та теорію Фазі, що аналізує неоднозначність поведінки оператора або водія.

Використовуючи сучасні засоби зв'язку можливо отримати на дисплеї машини інформацію про дорожні умови та іншу інформацію. Сьогодні також починають впроваджуватись навігаційні системи, що приймають сигнали глобальної системи визначення координат (GPS) через штучні супутники Землі, і визначають координати машини. Слід зазначити, що завдяки стрімкому розвитку мікропроцесорної техніки вона впроваджується як на автомобілях, так і на вантажних, підйомно-транспортних, будівельних та меліоративних машинах.

Наведені в посібнику структурні схеми мікропроцесорних систем керування агрегатами машин є універсальними і подібні до структурних схем аналогічних систем інших галузей.

Література

1. Бабич Н.П., Жуков И.А. Компьютерная схемотехника. Методы построения и проектирования: Учебное пособие.-К.: „МК-Прес".-2004.-576 с.
2. Боборыкин А.В., Липовецкий Г.П. и др. Однокристальные микроЭВМ. – М. : МИКАП, 1994. – 400с.
3. Буняк А.М Електроніка та мікросхемотехніка. – Київ – Тернопіль : СМП Астон, 2001. – 382с.
4. Вершинин О.Е., Применение микропроцесоров для автоматизации технологических процесов.–Л. : Энергоатомиздат. Ленингр. отд-ние, 1986.–208 с.
5. Гилмор Ч.М Введение в микропроцесорную технику. – М.: Мир, 1984. – 334 с.
6. Гончаренко С.У., Хаїмзон І.І. Учням про цифрову електроніку. – К.: Рад.шк., 1991.-173 с.
7. Гуртовцев А.Л., Гудыменко С.В. Программы для микропроцесоров : справочное пособие. – М.: Высшая школа,1989. – 352с.
8. Калабеков Б.А. Цифровые устройства и микропроцесорные системы. – М.:Телеком, 2000, -338 с.
9. Комп'ютерний словник/Пер. з англ.. В.О. Соловйова.-К.: Україна, 1997.-470 с.
10. Корнеев В.В. Киселёв А.В. Современные микропроцесоры. – М.: НОЛИДЖ, 1998, - 240 с.
11. Мартыненко И.И., А.П.Поддубный. Основы автоматики и микропроцесорной техники. – К.: Выща школа, 1988. – 248 с.
12. Сига Х., Мадзутани С. Введение в автомобильную электронику. М.: "Мир", 1989. – 232 с.
13. Токхайм Р.Л. Микропроцесоры : курс и упражнения. М.: Энергоатомиздат , 1988. – 336с.
14. Фрунзе А.В. Микроконтролери ? Это же просто ! Т.2.–М.: ООО "ИД Скимен". 2002.–392с.
15. Холленд Р.Ч. Микропроцессоры и операционные системы. –М.: Энергоатомиздат, 1991. – 192 с.
16. Шилейко А.В., Шилейко Т.И. Микропроцесоры.–М.: Радио и связь. 1986.–112с.
17. Якименко Ю.І., Терещенко Г.О та ін., за ред. Терещенко Г.О. Мікропроцесорна техніка : підручник, 2-ге вид., переробл. та доповн.- К.: ІВЦ "Видавництво "Політехніка"; "Кондор", 2004.–440с.

Перелік абревіатур

а) Українських:

АЛП – арифметико-логічний пристрій;

АСКІ – американський стандартний код обміну інформацією;

АСП – арифметичний співпроцесор;

АЦП – аналого-цифровий перетворювач;

БРГ – буферний регістр;

ВІС – велика інтегральна схема;

ГТІ – генератор тактових імпульсів;

ДДНФ – досконала диз'юнктивна нормальна форма логічної функції;

ДКНФ – досконала кон'юнктивна нормальна форма логічної функції;

ІВВ – інтерфейс введення/виведення;

КОП – код операції;

КШ – контролер шини керування;

МБК – мікропроцесорний блок керування;

МДН – польовий транзистор зі структурою „метал-діелектрик-напівпровідник";

МП – мікропроцесор;

МПС – мікропроцесорна система;

ОЗП – оперативний запам'ятовиючий пристрій;

ОМК – однокристальний мікроконтролер;

ПВВ – пристрій введення/виведення;

ПЗП – постійний запам'ятовуючий пристрій;

ППЗП – перепрограмований постійний запам'ятовуючий пристрій;

СВО – схема вихідної обробки сигналів;

СКС – система керування і синхронізації;

СП – система переривань;

СПО – схема попередньої обробки вхідних сигналів;

ЦАП – цифро-аналоговий перетворювач;

ША – шина адрес;

ШД – шина даних;

ШК – шина керування.

б) англійських:

A (Accumulutor) – акумулятор: спеціальний регістр мікропроцесора;

AB (Adress Bus) – адресна шина;

ADC (Analogue/Date Converter) – аналого-цифровий перетворювач;

CB (Control/Bus) – шина керування;

CISC (Complate Instruction Set Computing) – повний набір команд;

CP (Central Processor) – центральний процесор;

CS (Chip Select) – вибір кристалу;

DB (Date Bus) – шина даних;

DAC (Date/Analoque Converter) – цифро-аналоговий перетворювач;

DIO (Device Input/Output) – пристрій введення/виведення;

DPTR (Data Pointer Register) – регістр-вказівник даних;

EPROM (Erasable Programmable Read-Only Memory) – перепрограмований постійний запам'ятовуючий пристрій;

GnR (General Register) – регістр загального призначення;

PC (Program Counter) – лічильник команд;

P (Port) – порт введення/виведення;

PIO (Paralel Input/Output) – паралельне введення/виведення;

PSW (Processor State Word) – регістр слова стану процесора;

RAM (Random Access Memory) – оперативна пам'ять;

ROM (Read-Only Memory) – постійна пам'ять (тільки зчитування);

RISC (Reduced Instruction Set Computing) – обмежений набір команд;

SB (System Bus) – системна шина;

SP (Stach Pointer) – вказівник стеку;

T/C (Timer/Counter) – таймер/лічильник;

UART (Universal Asynchronous Receiver Transmitten) – універсальний асинхронний прийомопередавач.

Навчальне видання

Бочаров Сергій Юрійович

МІКРОПРОЦЕСОРНА ТЕХНІКА

Навчальний посібник

Друкується за авторською редакцією

Комп’ютерна верстка: Змієвський А.М.