МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЖАБІН В.І.
ЖУКОВ І.А.
ТКАЧЕНКО В.В.
КЛИМЕНКО І.А.

МІКРОПРОЦЕСОРНІ СИСТЕМИ

Навчальний посібник

Рекомендовано
Міністерством освіти і науки України
як навчальний посібник для студентів
вищих навчальних закладів,
які навчаються за напрямом підготовки
«Комп'ютерна інженерія»

Київ 2009

УДК 004.31 (076.5)

Укладачі:
ЖАБІН В.І., ЖУКОВ І.А., ТКАЧЕНКО В.В., КЛИМЕНКО І.А.

Рецензенти
В.Ф. ЄВДОКИМОВ, член.-кор. НАН України, д-р техн. наук, проф.
(Інститут проблем моделювання в енергетиці ім. Г.Є.Пухова НАН України)
І.А. ДИЧКА, д-р техн. наук, проф.
(Національний технічний університет України "КПІ")
Г.М. ЛУЦЬКИЙ, д-р техн. наук, проф.
(Національний технічний університет України "КПІ")

Гриф надано Міністерством освіти і науки України
(Лист№1/11 – 2022 від 01.04.09)

Мікропроцесорні системи: навчальний посібник/Уклад.:
Ж 752 В.І. Жабін, І.А. Жуков, В.В. Ткаченко, І.А. Клименко.-
К.:НАУ, 2009.- 492 с.

Містить лекційний матеріал, завдання, рекомендації і приклади до виконання практичних робіт, розрахунково-графічних робіт, курсового проектування. Надані задачі для самостійного розв'язування, типові завдання до рейтингового контролю.
Призначений для студентів напрямів «Комп'ютерна інженерія» та «Комп'ютерні науки».

ПЕРЕДМОВА

В навчальному посібнику узагальнені матеріали методичних розробок, які виконувалися авторами в процесі викладання курсів схемотехнічного напрямку на кафедрі комп'ютерних систем та мереж Інституту комп'ютерних технологій Національного авіаційного університету та кафедри обчислювальної техніки Національного технічного університету України «КПІ» для студентів навчального напряму «Комп'ютерна інженерія».

Матеріал навчального посібника присвячений вивченню принципів організації та дослідженню мікропроцесорних систем, особливостей архітектури, функціонування та системи команд мікропроцесорів у різних елементних базах, розробці програмного забезпечення з урахуванням цільової функції проектування.

Навчальний матеріал дисципліни структурований за модульним принципом і складається з трьох навчальних модулів. Окремим четвертим модулем є курсовий проект, який виконується в восьмому семестрі.

Крім необхідного теоретичного матеріалу, в посібнику приведені завдання та рекомендації до виконання практичних робіт. До практичних робіт надані теоретичні відомості, необхідні для виконання кожної роботи, приклади проектування, рекомендації до виконання завдання та саме завдання. До кожної практичної роботи надаються контрольні питання що застосовуються для контролю знань за відповідною тематикою.

Виконання практичних робіт дозволяє розширити і закріпити теоретичні знання з дисципліни, опанувати навички проектування і дослідження мікропроцесорних систем. Кожній практичній роботі повинна передувати самостійна підготовка студентів, в процесі якої вони докладно вивчають опис практичної роботи, відповідні розділи посібника, конспекту лекцій та літературні джерела. В процесі підготовки складається звіт про практичну роботу, в якому повинні бути відображені всі пункти теоретичного завдання, а також заготовлені для виконання експериментальної частини практичної роботи таблиці, алгоритми, схеми, осі для часових діаграм і таке інше. Перед початком практичної роботи результати підготовки перевіряються викладачем. Під час такої перевірки студент повинен представити заготовлений звіт і відповісти на контрольні питання. Перед початком наступного заняття в лабораторії студент представляє викладачеві цілком оформлений звіт за попередньою

роботою. Звіт повинен містити короткі теоретичні відомості, необхідні для виконання завдання, відповіді на контрольні питання, схеми, формули, алгоритми, таблиці, діаграми, графіки, отримані при виконанні завдання та в процесі експериментального дослідження схем, а також висновки. Залік за виконання практичної роботи студент одержує після співбесіди за тематикою виконаної роботи.

У посібнику надані завдання до виконання курсового проекту. Курсовий проект з дисципліни виконується з метою закріплення та поглиблення теоретичних знань та вмінь, набутих студентом у процесі засвоєння всього навчального матеріалу дисципліни в області проектування мікропроцесорних систем.

Конкретна мета курсового проекту є розробка мікропроцесорної системи на базі мікроконтролерів і однокристальних мікропроцесорів, а також програмного забезпечення для вирішення задач управління. Під час виконання курсового проекту студент отримає навички в розробці та налагодженні мікропроцесорних системи на основі заданої елементної бази, розробці програмного забезпечення з урахуванням заданої функції проектування, розробці структурних та принципових схем, користування довідковою літературою та оформлення проектно-конструкторської документації відповідно до діючих стандартів.

Теоретичний матеріал, зміст практичних робіт та курсового проекту безпосередньо відповідають навчальним планам дисциплін «Цифрові ЕОМ» та «Архітектура комп'ютерів». Посібник також може бути корисним при вивченні курсів «Комп'ютерна схемотехніка», «Проектування комп'ютерних систем» та інших курсів схемотехнічного напрямку.

Для проведення практичних робіт використовуються програмні засоби моделювання цифрових схем, які розроблені на кафедрі обчислювальної техніки НТУУ «КПІ» за участю студентів. В розробці програм моделювання ЕОМ приймали участь Р.Л. Антонов, О.Л. Брагинський, Д.В. Хабенко, Ю.В. Нечаєв, Г.Г. Салтищак і В.П. Щурко. Автори вдячні Є. Грицаю за допомогу в підготовці рукопису до видання.

Автори посібника вдячні рецензентам за слушні зауваження, що дозволили покращити матеріал посібника.

ЗМІСТ

ПРАКТИКУМ

ДОДАТКИ

ПЕРЕЛІК СКОРОЧЕНЬ

АЛБ – Арифметико-логічний блок
АЛП – Арифметико-логічний пристрій
БОД – Блок обробки даних
ВІС – Велика інтегральна схема
ЕОМ – Електронна обчислювальна машина
ЗП – Зовнішній пристрій
ЗПД – Зовнішня пам’ять даних
ЗПП – Зовнішня пам’ять програм
ІС – Інтегральна схема
КПДП – Контролер прямого доступу до пам’яті
КПП – Контролер пріоритетних переривань
МК – Мікрокоманда
МК48 – Мікроконтролер КР1816ВЕ48
МК51 – Мікроконтролер КР1816ВЕ51
МП – Мікропроцесор
МПС – Мікропроцесорна система
НВІС – Надвелика інтегральна схема
НОЗП – Надоперативний запам'ятовуючий пристрій
ОЗП – Оперативний запам’ятовуючий пристрій
ОП – Операційний пристрій
ПД – Пам’ять даних
ПЕ – Процесорний елемент
ПЗП – Постійний запам’ятовуючий пристрій
ПП – Пам’ять програм
ПУ – Пристрій управління
РЗП – Регістри загального призначення
РПД – Резидентна пам’ять даних
РПП – Резидентна пам’ять програм
СУ – Схема управління
СУСЗ – Схема управління станами та зсувами
СШ – Системна шина
ТТЛ – Транзисторно-транзисторна логіка
УС – Управляючі сигнали
ФАМ – Формувач адреси мікрокоманди
ША – Шина адреси
ШД – Шина даних
ШУ – Шина управління

ТЕОРЕТИЧНА ЧАСТИНА

1. ВВЕДЕННЯ В АРХІТЕКТУРУ МІКРОПРОЦЕСОРНИХ СИСТЕМ

1.1. Функціональна класифікація мікропроцесорів

Процесором (П) називається пристрій з програмним управлінням, що призначений для алгоритмічної обробки цифрової інформації. Процесор, реалізований у вигляді великої (ВІС) або надвеликої інтегральної мікросхеми (НВІС), називають *мікропроцесором* (МП).

Мікропроцесор відіграє важливу роль в цифрових системах різного призначення. Це системи обробки інформації, управління об'єктами і процесами, інформаційно-вимірювальні системи та інші, що застосовуються у промисловості та різних інших галузях.

За функціональними ознаками, що обумовлюють специфічні варіанти застосування в окремій області, можна виділити наступні класи мікропроцесорів: мікропроцесори *загального призначення* (*універсальні* мікропроцесори), *проблемна орієнтовані* та *спеціалізовані* мікропроцесори.

Серед проблемна орієнтованих мікропроцесорів найбільш широке розповсюдження отримали *мікроконтролери* та *цифрові сигнальні процесори*. До спеціалізованих належать мікропроцесори, що виготовлені для використання тільки в певних пристроях, наприклад, в пристроях мобільного зв'язку, медійних приладах і таке інше.

Мікропроцесори загального призначення використовуються для вирішення широкого кола задач обробки різноманітної інформації і знаходить застосування в персональних комп'ютерах, робочих станціях, серверах і інших цифрових системах масового застосування. Зазвичай це 32-, 64- та 120-розрядні мікропроцесори. У більшості випадків вони мають суперскалярну структуру, коли декілька операційних пристроїв виконують одночасну обробку даних. До найбільш відомих розробників мікропроцесорів цього класу належать компанії *INTEL*, *AMD* (*Advanced Micro Devise*), *MOTOROLA*, *SUN MICROSYSTEMS* та інші.

Мікроконтролери орієнтовані на рішення задач управління (*control* — управління). Мікроконтролери застосовуються для побудови приладів управління для різноманітних систем.

Особливістю мікроконтролерів є розміщення на одному кристалі, окрім центрального процесора, внутрішньої пам'яті і набору периферійних пристроїв. У склад периферійних пристроїв зазвичай входять паралельні та послідовні порти вводу-виводу даних, таймери, перетворювачі даних. У склад мікроконтролерів можуть входити інші пристрої, наприклад, блоки формування сигналів з широтно-імпульсною модуляцією, контролер рідкокристалічного дисплею. Завдяки використанню внутрішньої пам'яті і периферійних пристроїв системи управління, які реалізуються на базі мікроконтролерів, містять мінімальну кількість додаткових компонентів.

Найбільш простими і дешевими виробами цього класу є 8-розрядні мікроконтролери. Вони орієнтовані на використання у відносно нескладних пристроях масового випуску. Основними галузями їх використання являються промислова автоматика, автомобільна електроніка, вимірювальна техніка, побутова апаратура. У випадку необхідності є можливість додатково підключати зовнішню пам'ять команд і даних. Мікроконтролери цієї групи зазвичай виконують відносно невеликий набір команд (50 – 120), які використовують найбільш прості способи адресації.

Мікроконтролери з 16- і 32-розрядною сіткою характеризуються більш високою продуктивністю, розширеною системою команд і способів адресації, збільшеним набором регістрів і об'ємом пам'яті. Основні галузі застосування – складна промислова автоматика, телекомунікаційна апаратура, медична техніка. Тридцятидвохрозрядні мікроконтролери містять високопродуктивний процесор, який відповідає за своїми можливостями моделям мікропроцесорів загального призначення. Впровадження цих процесорів у склад мікроконтролерів дозволяє використовувати у відповідних системах управління величезний об'єм прикладного і системного програмного забезпечення, який був створений для персональних комп'ютерів. Деякі типи мікроконтролерів містять декілька виконавчих конвеєрів, які утворюють суперскалярну структуру.

До найбільш відомих розробників та виробників мікроконтролерів належать компанії *INTEL, ALTERA, ATMEL*

CORPORATION, *MICROCHIP TECHNOLOGY*, *MOTOROLA*, *NEC* (*Nippon Electronic Corporation*), *TEXAC INSTRUMENTS* та інші.

Цифрові сигнальні процесори призначені для цифрової обробки сигналів, як правило, у реальному часі. Прикладами цифрової обробки являються: фільтрація сигналів, пряме та обернене Фур'є-перетворення сигналів, обчислення кореляційної функції, згортка сигналів та ін. Набір команд цих процесорів містить спеціальні команди, що відсутні в інших процесорах, наприклад, множення з накопиченням *MAC* (*Multiplication with Accumulation*), що дозволяє ефективну обробку даних.

У відповідності з форматом подання відцифрованого аналогового сигналу ЦСП поділяються на процесори, які оброблюють числа з фіксованою або плаваючою точкою. Більш прості і дешеві ЦСП з фіксованою точкою оброблюють 16- або 24-розрядні операнди. Більша точність результатів досягається в ЦСП, що обробляють дані з плаваючою точкою, зазвичай з 32-розрядним форматом їх подання. Найбільш розповсюдженими є ЦСП сімейства *TMS* фірми *Texas Instruments*.

Технологія виготовлення мікропроцесорів всіх типів постійно розвивається, що забезпечує постійне зростання їх продуктивності та функціональних можливостей.

1.2. Різновиди архітектури мікропроцесорів

Під *архітектурою процесора* розуміють конфігурацію його основних програмних та апаратних компонентів з урахуванням їх можливостей та способів взаємодії.

Фактично це сукупність засобів, доступних користувачу (програмі), які забезпечують обробку цифрової інформації. Поняття архітектури включає набір програмно-доступних регістрів і операційних пристроїв, систему команд і способи адресації, об'єм і організацію пам'яті, засоби і способи обробки даних (обмін даними, переривання, прямий доступ до пам'яті та ін.).

Архітектура мікропроцесора тісно пов'язана з його структурою. Реалізація тих чи інших архітектурних особливостей потребує введення в структуру мікропроцесора відповідних пристроїв і забезпечення механізмів їх спільного функціонування.

Структура мікропроцесора визначає склад і зв'язок основних пристроїв і блоків, розміщених на його кристалі. У структуру мікропроцесорної системи входять:

– центральний процесор, що складається з пристрою управління та одного або декількох операційних пристроїв;

– внутрішня пам'ять (робочі регістри і регістри загального призначення, кеш-пам'ять);

– блоки оперативних та постійних запам'ятовуючих пристроїв;

– інтерфейси, що забезпечують обмін даними з пам'яттю та зовнішніми пристроями;

– периферійні пристрої (таймери, аналого-цифрові перетворювачі, спеціалізовані контролери);

– допоміжні схеми (генератор тактових імпульсів, схеми налагодження та тестування тощо).

За особливістю системи команд відрізняють наступні архітектури: *CISC, RISC* та *VLIW*.

CISC-архітектура визначається повним набором команд (*Complete Instruction Set Computer*). Вона реалізована у багатьох типах мікропроцесорів (наприклад *Pentium*), які виконують великий набір різноформатних команд з використанням численних способів адресації.

RISC-архітектура відповідає процесорам із скороченим набором команд (*Reduced Instruction Set Computer*). Поява *RISC*-архітектури обумовлена тим, що більшість *CISC*-команд і способів адресації використовуються досить рідко. Основна особливість *RISC*-архітектури проявляється у тому, що система команд складається з невеликої кількості часто використовуваних команд однакового формату, більшість яких можуть бути виконані за один командний цикл (такт) центрального процесора. Більш складні перетворення даних реалізуються на програмному рівні. Однак за рахунок значного підвищення швидкості виконання команд середня продуктивність *RISC*-процесорів може виявитися вище, ніж у процесорів з *CISC*-архітектурою, що залежить від класу задач.

Для скорочення кількості звернень до зовнішньої оперативної пам'яті *RISC*-процесори мають велику надоперативну внутрішню пам'ять. Звернення до зовнішньої пам'яті в *RISC*-процесорах використовується тільки в операціях завантаження даних до

внутрішніх регістрів загального призначення або пересилання результатів у пам'ять.

Завдяки вказаним достоїнствам у *CISC*-процесорах може використовуватися *RISC*-ядро. При цьому складні *CISC*-команди попередньо перетворюються у послідовність простих *RISC*-операцій і швидко виконуються *RISC*-ядром.

VLIW-архітектура належить до мікропроцесорів з використанням дуже довгих команд (*Very Large Instruction Word*). Окремі поля команди містять коди, які забезпечують виконання різноманітних операцій. Одна *VLIW*-команда може виконати одразу декілька операцій водночас у різних вузлах мікропроцесора. Формування «довгих» *VLIW*-команд виконує відповідних компілятор при трансляції програм, написаних на мові високого рівня. *VLIW*-архітектура реалізована в деяких типах сучасних мікропроцесорів і являється досить перспективною для створення надвисокопродуктивних процесорів.

За способом організації пам'яті та вибірки команд і даних розрізняють два основні класи архітектури. Розглянемо це на прикладі спрощених базових структур процесорів.

Архітектура фон-Неймана або *принстонська архітектура* (рис. 1.2), визначається використанням загальної основної (оперативної) пам'яті для зберігання програм і даних, що дозволяє оперативно і ефективно перерозподіляти її об'єм в залежності від вирішуваних задач в кожному конкретному випадку.

Гарвардська архітектура відрізняється фізичним розподілом пам'яті на окрему пам'ять команд (як правило, постійну) і окрему оперативну пам'ять даних (рис. 1.3). Завдяки розділу потоків команд і даних, а також об'єднанню операцій звернення до різних модулів пам'яті забезпечується більш висока продуктивність, ніж при використанні загальної пам'яті для команд і даних.

Гарвардська архітектура отримала широке застосування в мікроконтролерах, для яких можливо заздалегідь записати в постійну пам'ять всі необхідні програми управління. Зауважимо, що розподіл пам'яті застосовується також у внутрішній структурі сучасних високопродуктивних універсальних мікропроцесорів, в яких використовують кеш-пам'ять окремо для команд і даних.

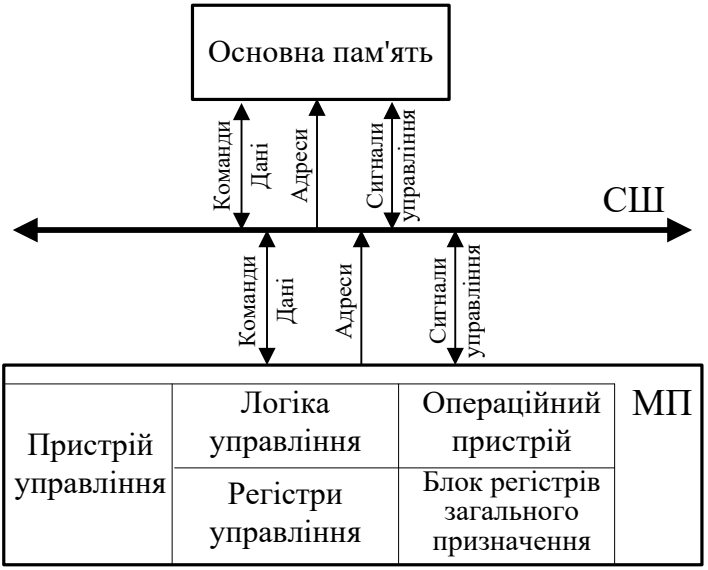


Рис. 1.2. Архітектура фон-Неймана

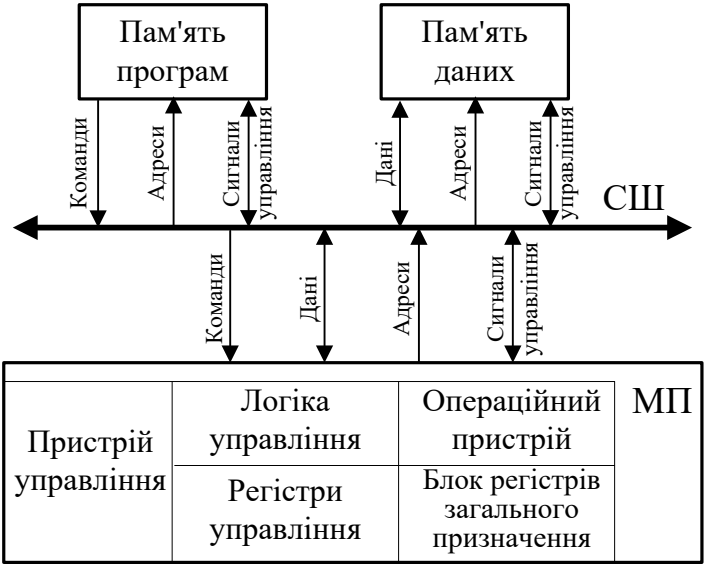


Рис.1.3. Гарвардська архітектура

1.3. Організація обчислювальних процесів в мікропроцесорних системах

Мікропроцесорна система (МПС) є функціонально закінченим виробом, що складається з декількох цифрових пристроїв, включаючи мікропроцесор. Організацію обробки даних розглянемо на прикладі мікропроцесорної системи, що відповідає базовій архітектурі фон-Неймана (рис. 1.4).

Мікропроцесорні системи у своїй більшості мають модульну структуру. Окремі пристрої (модулі), обмінюються даними через системну шину (СШ) – загальну магістраль обміну даними. Основним модулем є мікропроцесор, що вміщує пристрій управління (ПУ), операційний пристрій (ОП), блок робочих регістрів та регістрів загального призначення , що є внутрішнім надоперативним запам'ятовуючим пристроєм (НОЗП). ОП виконує арифметичні та логічні операції над даними, ПУ забезпечує програмне управління процесом обробки інформації, організовує взаємодію всіх пристроїв.

Основна пам'ять зазвичай реалізується у вигляді модулів оперативного запам'ятовуючого пристрою (ОЗП) та постійного запам'ятовуючого пристрою (ПЗП). ОЗП застосовується для зберігання програми і даних, що обробляються. У сучасних мікропроцесорних системах (персональних комп'ютерах, серверах, робочих станціях) ємність ОЗП досягає сотні Мбайт. Для зменшення часових витрат під час звернення до СШ у склад більшості сучасних мікропроцесорів вводиться швидкодіюча проміжна пам'ять – КЕШ-пам'ять, що має обмежену ємність (до декількох сотень Кбайт). ПЗП застосовується для збереження констант та незмінних програм, в тому числі, модулі системного програмного забезпечення.

Зовнішні пристрої (ЗП) підключаються до системи за допомогою різних інтерфейсів із паралельними або послідовними протоколами обміну даними. До ЗП належать клавіатура, монітор, зовнішні носії інформації, датчики і перетворювачі інформації, різноманітні виконуючи пристрої.

СШ складається з декілька десятків (у складних системах – декількох сотень) провідників, що за функціональним призначенням поділяються на шину даних (ШД), шину адреси (ША) та шину управління (ШУ). Адреси і дані можуть передаватися по загальній сумісній шині адреси/даних (ШАД). В цьому випадку передача

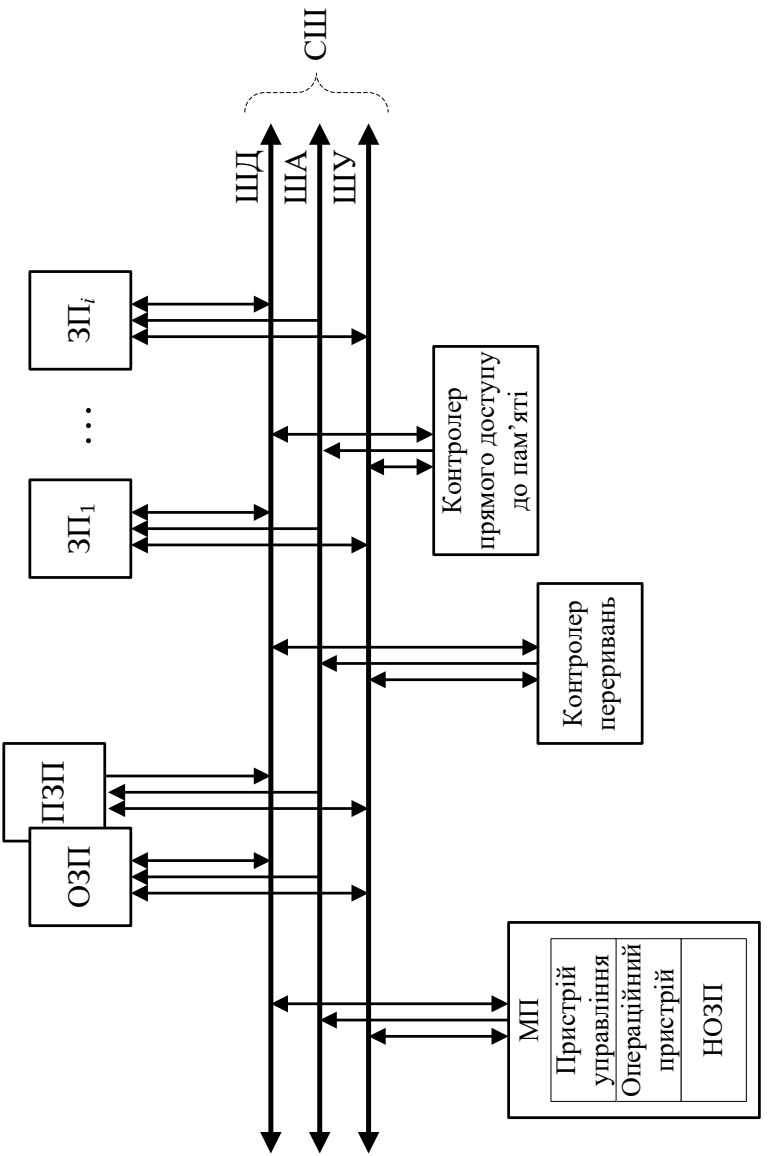


Рис. 1.4. Структура мікропроцесорної системи

адреси і даних розділяється у часі. Команди вибираються у пристрій управління мікропроцесора, а дані оброблюються в операційному пристрої. ШУ забезпечує передачу управляючих сигналів.

Для реалізації різних режимів роботи до мікропроцесорної системи підключають додаткові пристрої: контролери переривань, контролери прямого доступу до пам'яті та інші.

Основні режими роботи мікропроцесорної системи:

- виконання основної програми;
- звернення до підпрограми;
- обслуговування переривань;
- реалізація прямого доступу до пам'яті.

Виконання основної програми. Виконувана програма завантажується до оперативної пам'яті. У процесі виконання основної програми центральний процесор зчитує з пам'яті і послідовно виконує команди в певній послідовності. Основний цикл виконання команди складається з наступних етапів:

- вибірка команди з основної пам'яті;
- розпаковка (дешифрування) команди;
- виконання команди;
- формування адреси наступної команди.

Звернення до підпрограм забезпечується за допомогою спеціальної команди звернення до підпрограми, яка вказує адресу першої команди підпрограми, що викликається. Перед переходом на підпрограму виконується процедура збереження адреси повернення в основну програму у спеціальному регістрі або у стеку. Повернення до основної програми відбувається під час виконання спеціальної команди, що завершує підпрограму, при цьому збережена адреса повернення забезпечує подальше виконання основної програми.

Обслуговування переривань. В процесі обробки інформації можуть виникати ситуації, коли необхідно перервати виконання основної програми і перейти на підпрограму обробки ситуації, що викликала переривання. У таких ситуаціях мікропроцесор завершує виконання чергової команди, зберігає адресу повернення і стан програми, що переривається, та переходить на виконання програми обслуговування переривань.

Для забезпечення пріоритетного обслуговування запитів від багатьох зовнішніх пристроїв застосовуються спеціальні мікросхеми

– контролери пріоритетних переривань. Деякі типи мікропроцесорів мають убудовані пристрої для обслуговування переривань.

Режим прямого доступу до пам'яті. Для збільшення швидкості обміну даними між пристроями системи використовується режим прямого доступу до пам'яті, який реалізується за допомогою контролера прямого доступу до пам'яті. Контролер ініціює запит на захват шин для пересилання даних між пристроями системи. Попередню ініціалізацію контролера виконує процесор, для чого пересилає по шині даних в контролер прямого доступу інформацію, необхідну для управління обміном (адресу комірки пам'яті, в якій розміщується перший байт даних, що записуються або зчитуються, адресу порту, загальну кількість даних, що передаються, напрям передачі та інше). Процесор по запиту відключається від шини даних і шини адреси, надаючи їх контролеру для організації обміну. Після завершення обміну центральний процесор отримує сигнал від контролера і переходить до виконання основної програми.

Більш детальніше режими роботи процесора будуть розглянуті нижче.

2. МІКРОПРОЦЕСОРИ НА ОСНОВІ СЕКЦІОНОВАНИХ ІНТЕГРАЛЬНИХ СХЕМ

2.1. Загальні відомості

Секціоновані мікропроцесорні серії інтегральних схем (ІС) призначені для розробки МПС з будь-якою системою команд і довільною розрядністю. Розгляд таких комплектів доцільний для вивчення архітектурних особливостей процесорів і мікропроцесорних систем.

Особливості секціонованих мікропроцесорних комплектів розглянемо на прикладі серії К1804, яка широко використовувалась до переходу на технологію побудови систем на заказних ВІС та ПЛІС. Комплект складається з наступних ІС:

К1804ГГ1	– генератор тактових сигналів
К1804ВА1, К1804ВА2, К1804ВАЗ	– чотирирозрядні приємопередавачі;
К1804ВЖ1	– шістнадцятирозрядна схема виправлення помилок в ЗП;
К1804ВН1	– блок пріоритетного переривання на 8 запитів;
К1804ВР1	– схема прискореного переносу;
К1804ВР2	– блок управління станами та зсувами;
К1804ВР3	– розширювач блока пріоритетного переривання;
К1804ВС1, К1804ВС2	– чотирирозрядні процесорні елементи;
К1804ВУ1, К1804ВУ2	– чотирирозрядні генератори адрес мікрокоманд;
К1804ВУ3	– схема управління генератором адрес мікрокоманд;
К1804ВУ4	– дванадцятирозрядний генератор адрес мікрокоманд (формувач адреси наступної мікрокоманди);
К1804ВУ5	– чотирирозрядний генератор адрес мікрокоманд;
К1804ИР1	– чотирирозрядний регістр;
К1804ИР2, К1804ИР3	– восьмирозрядні регістри.

Розглянемо основні ІС серії К1804 та приведемо особливості проектування мікропроцесорних систем на їх основі.

2.2. Процесорний елемент

ІС К1804ВС1 виконана за технологією ТТЛШ і є чотирирозрядним універсальним процесорним елементом (ПЕ), призначеним для побудови блоків обробки даних. Структурна схема ПЕ зображена на рис. 2.1.

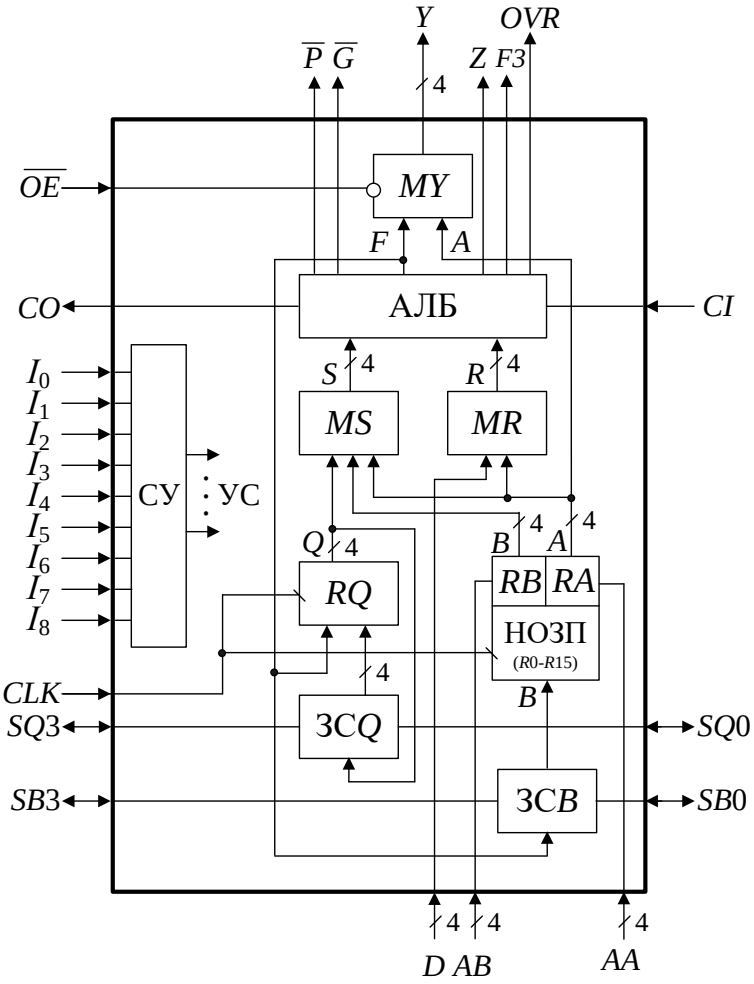


Рис. 2.1. Структурна схема процесорного елементу ІС К1804ВС1

Процесорний елемент К1804ВС1 складається з наступних функціональних частин:

АЛБ	– арифметико-логічний блок;
НОЗП	– надоперативний запам'ятовуючий пристрій, що складається з шістнадцяти регістрів $R15 - R0$;
RA, *RB*	– регістри тимчасового зберігання адрес операндів за каналом *A* і *B*;
ЗС*Q*, ЗС*B*	– зсувачі;
MS, *MR*, *MY*	– мультиплексори вибору операндів *S i R* та даних *Y*
СУ	– схема управління.

Умовне позначення ПЕ зображене на рис 2.2, призначення виводів мікросхеми описане в табл. 2.1.

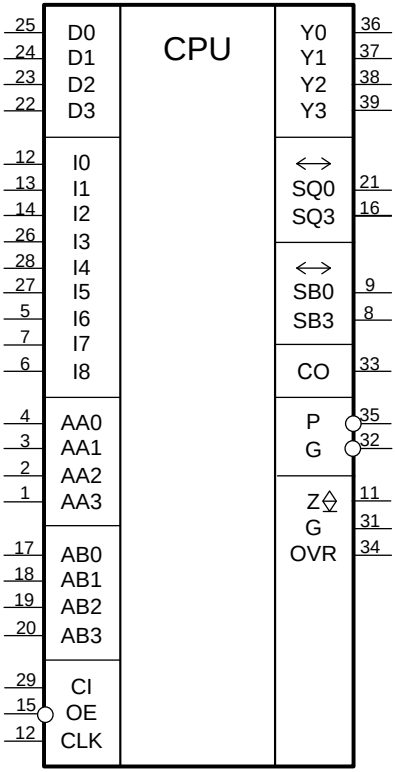


Рис 2.2. Умовне графічне позначення мікросхеми ПЕ К1804ВС1

Молодший розряд ПЕ має номер 0, а старший – номер 3.

Виходи *Y* мають три стани (0, 1 і високоомний). Встановлення в високоомний стан цих виходів здійснюється подачею на вхід *OE* одиничного потенціалу $\overline{OE} = 1$.

Виводи *SQ*3, *SQ*0, *SB*3 і *SB*0 є двоспрямованими і мають також три стани. При виконанні на зсувачах ЗС*B* і ЗС*Q* зсуву вправо виводи *SB*3 і *SQ*3 є входами, а виводи *SB*0 і *SQ*0 – виходами. При зсуві інформації вліво функції виводів змінюються на протилежні. Якщо зсув на зсувачі не виконується, то відповідні йому виводи знаходяться в високоомному стані. Вихід *Z* виконаний за схемою з відкритим колектором. Решта виходів *IC* мають два стани (0 та 1).

Таблиця 2.1. Функціональне призначення виводів мікросхеми процесорного елементу К1804ВС1

Номери виводів	**Позначення**	**Функціональне призначення**
1 – 4	*AA*3 – *AA*0	Входи адрес регістрів НОЗП за каналом *A*
12 – 14, 26, 28, 27, 5 – 7	*I*8 – *I*0	Входи мікрокоманди
8 (9)	*SB*3 (*SB*0)	Вхід (вихід) старшого (молодшого) розряду ЗС*B*
11	*Z*	Вихід ознаки нульового результату в АЛБ
15	*CLK*	Вхід синхросигналів
16 (21)	*SQ*3 (*SQ*0)	Вхід (вихід) старшого (молодшого) розряду ЗС*Q*.
17-20	*AB*0 – *AB*3	Входи адреси регістрів НОЗП за каналом *B*
22 – 25	*D*3 – *D*0	Входи даних
29	*CI*	Вхід переносу в АЛБ
31	*F*3	Вихід старшого розряду АЛБ
32,35	*G*,*P*	Виходи прискореного переносу
33	*CO*	Вихід переносу з АЛБ
34	*OVR*	Вихід ознаки переповнювання результату
36 – 39	*Y*0 – *Y*3	Виходи даних
40	*OE*	Вхід дозволу видачі даних
33	*CO*	Вихід переносу з АЛБ

В ПЕ використовується двоадресний НОЗП, який забезпечує видачу інформації за двома незалежними каналами *A* і *B* (рис. 2.1.). Інформація на виходах *A* і *B* визначається відповідно адресами *AA* і *AB*. Якщо на входах *AA* і *AB* встановлені однакові адреси регістрів, то на виходи *A* і *B* видається вміст одного й того самого регістру. Адреси регістрів дорівнюють двійковим еквівалентам їх номерів. Наприклад, регістр *R*5 має адресу 0101 і таке інше. Запис інформації в НОЗП відбувається за адресою *B* – тільки за одним каналом.

Для управління ПЕ в структурі мікрокоманди відведено 18 розрядів (рис. 2.3).

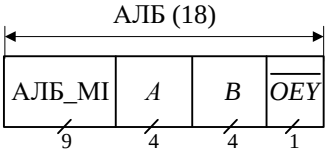


Рис. 2.3. Поля мікрокоманди, які використовуються для управління ПЕ

Характер перетворення інформації в кожнім такті роботи ПЕ визначається інформаційним словом (мікроінструкцією МІ), що розміщується в полі АЛБ_МІ і надходить на входи $I8-I0$ мікросхеми К1804ВС1. Формат інформаційного слова для управління ПЕ наведений на рис. 2.4.

Рис. 2.4. Формат інформаційного слова АЛБ_МІ мікрокоманди

Поле ($I_8I_7I_6$) управляє мультиплексором *MY*, зсувачами З*CQ* і З*CB*, а також вибором приймача результату (рис.2.1). Кодування розрядів даного поля наведене у табл. 2.2.

Результат *F* мікрооперації, що виконувалась в АЛБ, може бути записаний в НОЗП за каналом *B* ($F \to B$) без змін, а також у модифікованому вигляді – із зсувом на один розряд вліво ($2F \to B$)

або вправо ($F/2 \to B$). За встановлення значення $I_8I_7I_6 = 000$ результат записується в регістр *RQ* ($F \to Q$). Результат не фіксується в регістрах ПЕ за встановлення значення $I_8I_7I_6 = 001$.

Таблиця 2.2. Кодування поля управління мультиплексором

Розряди мікрокоманди			Мікрооперації		*Y*
I_8	I_7	I_6	НОЗП	*RQ*	
0	0	0	–	$F \to Q$	*F*
0	0	1	–	–	*F*
0	1	0	$F \to B$	–	*A*
0	1	1	$F \to B$	–	*F*
1	0	0	$F/2 \to B$	$Q/2 \to Q$	*F*
1	0	1	$F/2 \to B$	–	*F*
1	1	0	$2F \to B$	$2Q \to Q$	*F*
1	1	1	$2F \to B$	–	*F*

Під час встановлення на вході сигналу дозволу видачі даних *OE* ($\overline{OE} = 1$) та за $I_8I_7I_6 = 010$ мультиплексор *MY* видає на вихідні шини *Y* інформацію з виходів каналу *A* НОЗП (з регістру *RA*), а в інших випадках – з виходів *F* АЛБ.

Одночасно із записом результату в НОЗП в регістрі *RQ* може бути виконаний зсув інформації вліво ($I_8I_7I_6 = 110$) або вправо ($I_8I_7I_6 = 100$).

Розряди $I_5I_4I_3$ поля АЛБ_МІ МК (рис.2.4) визначають мікрооперацію в АЛБ відповідно до табл. 2.3, де *R* і *S* виходи мультиплексорів *MR* і *MS*.

Мікрооперації підсумовування і віднімання виконуються у доповнювальному коді з урахуванням вхідного переносу *CI*. Логічні мікрооперації – диз'юнкція (АБО), кон'юнкція (І) і сума за модулем два (ВИКЛЮЧНЕ АБО) є порозрядними. Вибір джерел операндів здійснюється за допомогою мультиплексорів *MS* і *MR* та управляється розрядами $I_2I_1I_0$ поля АЛБ_МІ мікрокоманди (табл. 2.4).

Таблиця 2.3. Кодування поля $I_5\ I_4\ I_3$

Розряди мікрокоманди			**Мікрооперація в АЛБ**
I_5	I_4	I_3	
0	0	0	$R+S+CI$
0	0	1	$S-R-1+CI$
0	1	0	$S-R-1+CI$
0	1	1	$R \vee S$
1	0	0	$R \& S$
1	0	1	$\overline{R} \& S$
1	1	0	$R \oplus S$
1	1	1	$\overline{R \oplus S}$

Таблиця 2.4. Кодування поля $I_2\ I_1\ I_0$

Розряди мікрокоманди			Джерела операндів	
I_2	I_1	I_0	R	S
0	0	0	A	Q
0	0	1	A	B
0	1	0	0	Q
0	1	1	0	B
1	0	0	0	A
1	0	1	D	A
1	1	0	D	Q
1	1	1	D	0

Такт роботи ПЕ полягає в наступному.

За додатним перепадом *CLK* на управляючі входи ПЕ подається слово мікрокоманди (рис. 2.1). В регістри *RA* і *RB* з відповідних регістрів НОЗП, які визначаються адресами на входах *AA* і *AB*, записуються дані. Відповідно до значень розрядів $I_8 - I_0$

мікрокоманди схема управління (СУ) виробляє необхідні управляючі сигнали (УС), в результаті цього обираються джерела операндів. Далі відбувається перетворення операндів в АЛБ і видається результат на вихід *F*. Якщо мультиплексор *MY* відкритий ($\overline{OE} = 0$), результат видається на шину *Y*. Залежно від результату, одержуваного в АЛБ, на відповідних виходах ПЕ формуються ознаки *Z*, *F*3, *OVR*.

Одиничний сигнал на виході *Z* відповідає нульовому значенню результату. Ознакою арифметичного переповнювання в АЛБ є одиничний сигнал на виході *OVR*. Вивід *F*3 є виходом старшого розряду АЛБ і використовується для аналізу знаку результату.

За від'ємним перепадом синхросигналу $CLK = 0$ результат фіксується у вибраному приймачеві результату – в регістрах НОЗП або в регістрі *RQ*. Для забезпечення правильної роботи ПЕ необхідно, щоб сигнали на управляючих входах ПЕ не змінювали свого значення за $CLK = 0$.

Часові параметри ІС К1804ВС1 наведені в табл. 2.5.

Таблиця 2.5. Часові параметри ІС К1804ВС1

Параметри ІС К1804ВС1	**Значення, нс**	**Параметри ІС К1804ВС1**	**Значення, нс**
Тривалість сигналу		від *I* до *Y*, *CO*, *F*3	50
$CLK = O$ (*min*)	30	від *I* до *G*, *P*	42
Час попереднього		від *I* до *Z*	65
встановлення відносно		від *I* до *OVR*	59
додатного перепаду *CLK*		від *I* до *SB*3, *SB*0	70
сигналів на входах (*min*):		від *D* до *Y*	39
RA, *RB*	93	від *B* до *G*, *D*	31
D, *I*	70	від *D* до *Z*	55
CI	55	від *D* до *CO*, *F*3	41
*SQ*3, *SQ*0, *SB*3, *SB*0	20	від *D* до *OVR*	45
Час затримки		від *D* до *SB*3, *SB*0	53
від *RA*, *RB* до у, *F*3	75	від *CI* до *Y*	27
від *RA*, *RB* до *G*, *P*	59	від *C1* до *Z*	46
від *RA*, *RB* до *Z*	85	від *CI* до *CO*	20
від *RA*, *RB* до *CO*	70	від *CI* до *F*3	24
від *RA*, *RB* до *OYR*	76	від *CI* до *OVR*	26
	90	від *CI* до *SB*3, *SB*0	45

від *RA*, *RB* до *SB*3, *SB*0			

Приклад 2.1. Розробити мікропрограму для процесорного елементу, що реалізує заданий мікроалгоритм (рис. 2.5).

Вихідні дані: $X1=-3$, $X2=-9$, $X3=11$.

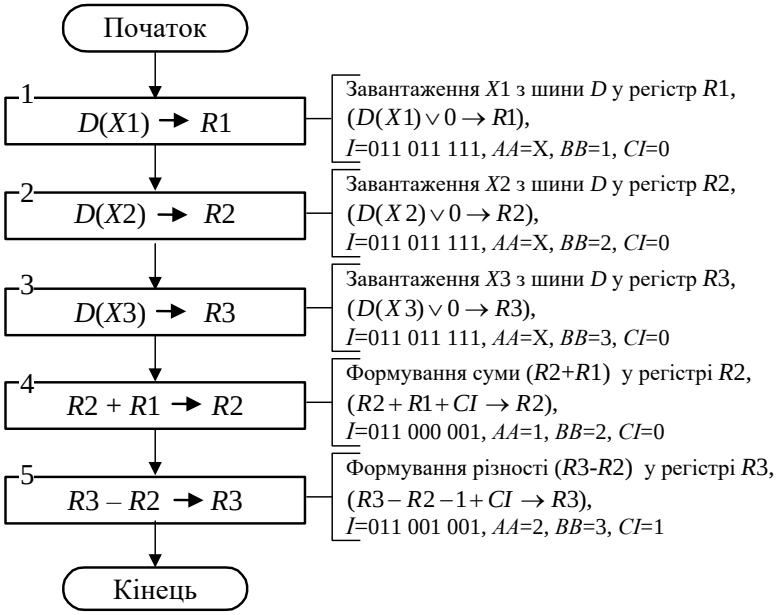


Рис. 2.5. Вихідний мікроалгоритм

Вважатимемо, що для подання операндів використовується доповнювальний код. В коментарях до кожної операторної вершини (рис. 2.5) наведені значення управляючих сигналів АЛБ_МІ, *AA*, *BB*, *CI*, що забезпечують виконання відповідної мікрооперації.

Послідовність мікрокоманд, що реалізують заданий мікроалгоритм, відображена у табл. 2.6. Цифрова діаграма стану регістрів показана у табл. 2.7.

Розглянемо мікрооперацію завантаження $D(X1) \to R1$, що виконується в першому такті. Зазначимо, що дані в регістри НОЗП

можуть бути завантажені лише, як результат виконання мікрооперації в АЛБ. Для завантаження даних з шини даних виконаємо мікрооперацію логічного додавання. Джерелами даних, що надходять на входи *S* і *R* АЛБ (рис. 2.1), є машинний нуль і дані на вхідній шині *D* ($D(X1)\vee 0\to R1$) Відповідно до табл. 2.4. визначаємо розряди $I_2I_1I_0$ поля АЛБ_МС мікрокоманди $I_2I_1I_0=111$. Мікрооперація логічне І, що здійснюється у цьому такті, визначається розрядами $I_5I_4I_3=011$ поля АЛБ_МС мікрокоманди (табл. 2.3). Результат, отриманий на виході *Y* АЛБ записується в регістр НОЗП за адресою, виставленою на шині *AB*. За умовою завдання результат має бути записаний у регістр *R*1, тому у полі *B* мікрокоманди (рис. 2.9) розміщується адреса відповідного регістру у шістнадцятирічному поданні *B* = 1. При цьому кодуємо розряди $I_8I_7I_6$ поля АЛБ_МС мікрокоманди (табл. 2.2), отримуємо $I_8I_7I_6=011$. Мікрооперація додавання, що здійснюється у четвертому такті, визначається розрядами $I_5I_4I_3=000$ поля АЛБ_МІ мікрокоманди (табл. 2.3). при цьому встановлюється *C1*=0, що відповідає розрядам $I_{12}I_{11}=00$ поля СУСЗ_МІ мікрокоманди (табл. 2.19). У полях *A* і *B* мікрокоманди (рис. 2.3) розміщуються номери регістрів НОЗП у шістнадцятирічному поданні, що є джерелами операндів. Результат, отриманий на виході *Y* АЛБ записується в регістр НОЗП за адресою, виставленою на шині *AB*, тому у полі *B* мікрокоманди (рис. 2.3) розміщується номер регістра-приймача результату (*A* = 1, *B* = 2). Мікрооперація віднімання, що здійснюється у п'ятому такті, визначається розрядами $I_5I_4I_3=001$ поля АЛБ_МІ мікрокоманди (табл. 2.3). при цьому встановлюється *C1*=1, що відповідає розрядам $I_{12}I_{11}=01$ поля СУСЗ_МІ мікрокоманди (табл. 2.19).

Наведемо мікропрограму у мнемонічних кодах мікроасемблеру:

```
        /-- Завантаження даних в регістри НОЗП---
0000    accept r1: 0fffdh /--(–3)ДК → R1
0001    accept r2: 0fff7h /--(–9)ДК → R1
0002    accept r3: 0000bh /--(11)ДК → R1
        /--Область програми----------------------
0003    {add r2,r2,r1,z;}  /--Обчислення суми
```

```
0004    {sub r3,r2,r3,nz;} /--Обчислення різності
        {}
```

Таблиця 2.6. Кодова карта

№ так-ту	Адреса [16]	Константа БОД		БОД													
				ВС1 (АЛБ)				ВР2 (СУСЗ)									
		D [16]	$\overline{OED}$	АЛБ_МІ [2]	А [16]	В [16]	$\overline{OEY}$	СУСЗ_МІ [2]	$\overline{EC}$	$\overline{EZ}$	$\overline{EN}$	$\overline{EV}$	$\overline{CEN}$	$\overline{CEM}$	$\overline{OECT}$	$\overline{SE}$	
1	0000	*FFFD*	0	011 011 111	*	1	0	00 00000 000000	0	0	0	0	1	1	1	1	
2	0001	*FFF7*	0	011 011 111	*	2	0	00 00000 000000	0	0	0	0	1	1	1	1	
3	0002	000*B*	0	011 011 111	*	3	0	00 00000 000000	0	0	0	0	1	1	1	1	
4	0003	****	1	011 000 001	1	2	0	00 00000 000000	0	0	0	0	1	1	1	1	
5	0004	****	1	011 001 001	2	3	0	01 00000 000000	0	0	0	0	1	1	1	1	

Таблиця 2.7. Цифрова діаграма стану регістрів

№ так-ту	Вихідний стан				Інформація у приймачі результату, [16]	Мікрооперація
	Шина ***D***, [16]	***R*1**, [16]	***R*2**, [16]	***R*3**, [16]		
1	*FFFD*	****	****	****	*R*1:= *FFFD*	$R1 := D(FFFD) \vee 0$
2	*FFF*7	*FFFD*	****	****	*R*2:= *FFF*7	$R2 := D(FFF7) \vee 0$
3	000*b*	*FFFD*	*FFF*7	****	*R*3:= 000*B*	$R3 := D(000B) \vee 0$
4	****	*FFFD*	*FFF*7	000*B*	*R*2:= *FFF*4	$R2 := R2 + R1 + 0(CI)$
5	****	*FFFD*	*FFF*4	000*B*	*R*3:= 0017	$R3 := R2 - R3 - 1 + 1(CI)$

2.3. Схема управління станами та зсувами

2.3.1. Загальна структура та принцип функціонування

Схема управління станами та зсувами (СУСЗ) призначена для спільної роботи з ПЕ К1804ВС1, разом з якими складає блок обробки даних (БОД). Схема управління станами та зсувами реалізована на ІС К1804ВР2, структурна схема якої наведена на рис. 2.6. Загальна структура БОД та поєднання виводів мікросхем К1804ВР2 та наведені на рис. 2.7.

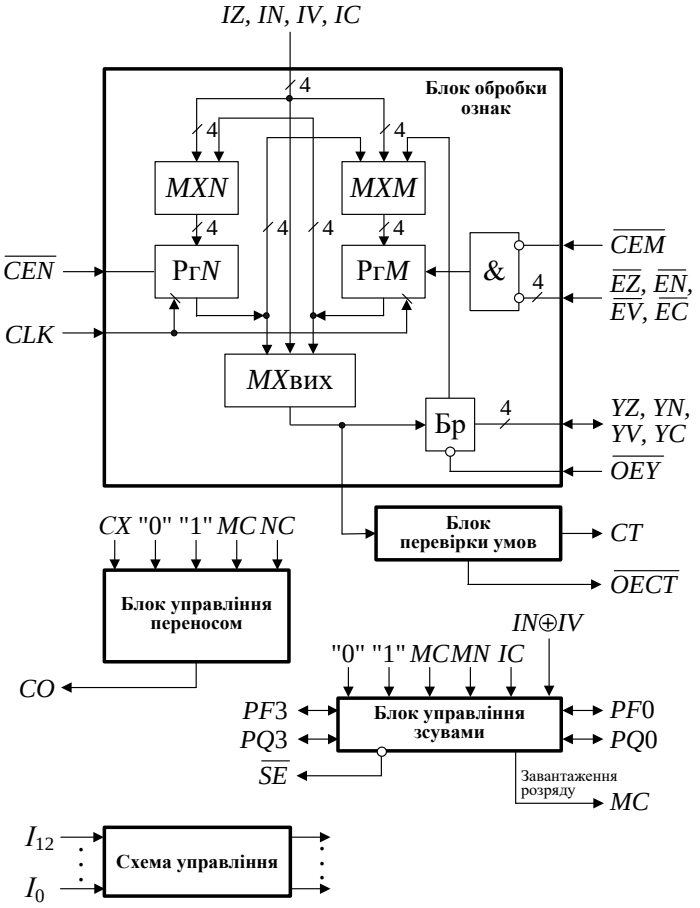


Рис. 2.6. Структурна схема СУСЗ ІС К1804ВР2

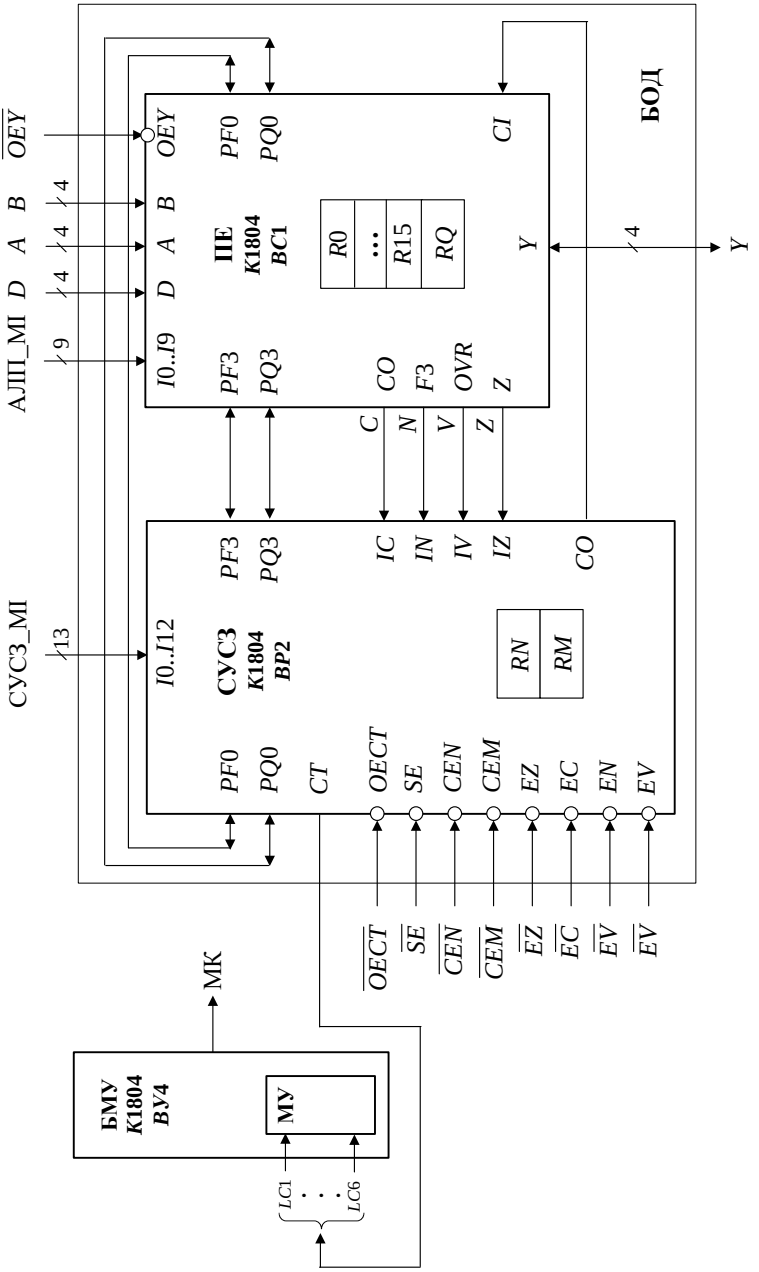


Рис. 2.7. Блок обробки даних

Для управління БОД у структурі мікрокоманди відведено 43 розряди. На рис. 2.8 зображені відповідні поля МК.

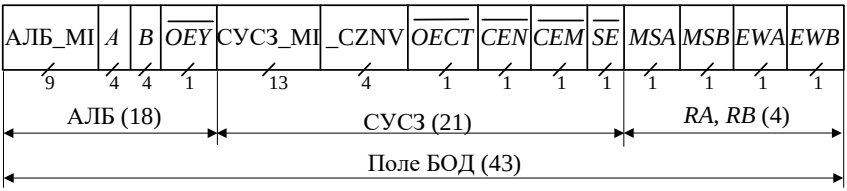


Рис. 2.8. Поля мікрокоманди, які використовуються для управління блоком обробки даних

Умовне графічне позначення ІС К1804ВР2 та нумерація виводів приведені на рис. 2.9, а їх функціональне призначення – в табл. 2.7.

Таблиця 2.7. Функціональне призначення виводів ІС К1804ВР2

Позначення	**Назва**	**Призначення**
$I12 - I0$	Інформаційні входи МК	Визначають операцію, що виконується СУСЗ
IC, IN, IV, IZ	Входи ознак стану	Використовуються для передачі ознак стану АЛБ (переносу, знаку, переповнювання, рівність нулю) в СУСЗ
$\overline{CEM}$	Вхід дозволу запису ознак в Рг*M*	При значенні 0 на вході $\overline{CEM}$ ($\overline{CEM} = 0$) запис в Рг*M* дозволений, інакше ($\overline{CEM} = 1$) – заборонений
$\overline{CEN}$	Вхід дозволу запису ознак в Рг*N*	При значенні 0 на вході $\overline{CEN}$ ($\overline{CEN} = 0$) запис в Рг*N* дозволений, інакше ($\overline{CEN} = 1$) – заборонений
$\overline{EC}$, $\overline{EN}$, $\overline{EV}$, $\overline{EZ}$	Входи дозволу запису ознак	При значенні 1 на входах $\overline{EC}$, $\overline{EN}$, $\overline{EV}$, $\overline{EZ}$ запис у відповідні розряди Рг*M* заборонений, а при значенні 0 – дозволений, якщо одночасно встановлений сигнал дозволу запису $\overline{CEM} = 0$

YC, *YN*, *YV*, *YZ*	Двоспрямована шина даних	Застосовується як вхід даних під час запису в регістр Рг*M* і як вихід даних під час виводу інформації з регістрів Рг*N*, Рг*M* або з входів *IC*, *IN*, *IV*, *IZ* СУСЗ
$\overline{OEY}$	Вхід управляючого сигналу дозволу виводу інформації	За значення 0 на вході $\overline{OEY}$ ($\overline{OEY}=0$) дозволений вивід інформації зі СУСЗ через шину *Y*. При $\overline{OEY}=1$ шина *Y* знаходиться в стані високого опору – вивід даних на шину *Y* заборонений
$\overline{OECT}$	Вхід управляючого сигналу дозволу формування коду умови	За значення 0 на вході $\overline{OECT}$ ($\overline{OECT}=0$) дозволений вивід коду умови на вихід *CT*. Якщо $\overline{OECT}=1$, формування коду умови на виході *CT* заборонене
CT	Вихід умови	Результат перевірки умови
*PF*0, *PF*3, *PQ*0, *PQ*3	Двоспрямовані виводи зсуву	Призначені для організації зсувів в МПС шляхом поєднання їх із відповідними виводами зсувів
$\overline{SE}$	Вхід дозволу зсуву	За значення 0 на вході $\overline{SE}$ ($\overline{SE}=0$) виконання зсувів дозволено, інакше ($\overline{SE}=1$) – заборонено, під час цього виводи зсуву СУСЗ знаходяться в стані високого опору
CO	Вихід формування переносу	Сформований на виході *CO* сигнал використовується в якості вхідного переносу *CI* АЛБ
CX	Вхід формування переносу	Використовується як одне з джерел при формуванні сигналу вихідного переносу *CO*
CLK	Тактовий вхід	Запис інформації в регістр стану відбувається за додатним перепадом тактового сигналу *CLK*. Решта схем є комбінаційними – їх функціонування не залежать від значення сигналу *CLK*

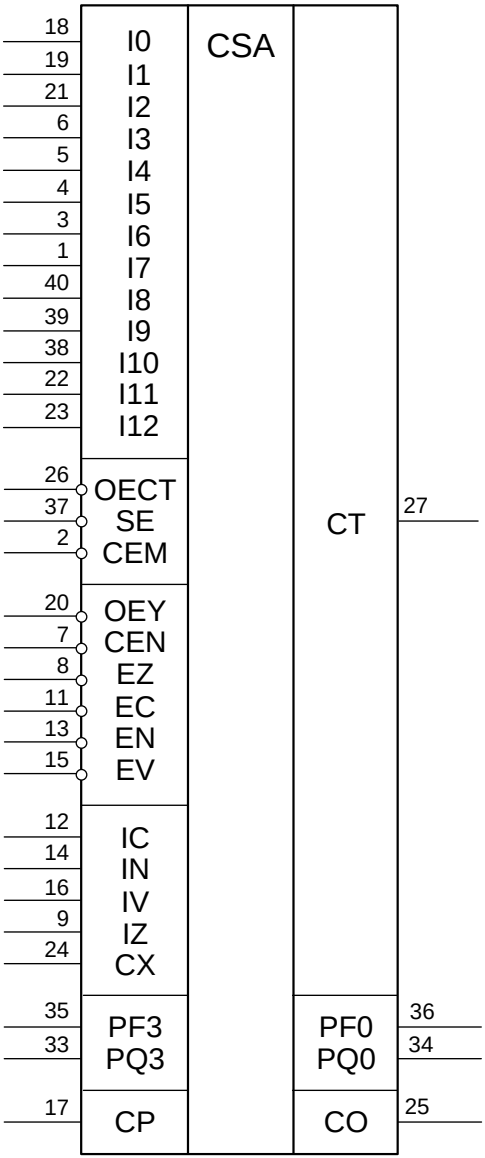


Рис. 2.9. Умовне графічне позначення ІС К1804ВР2

Функціональні блоки, що входять до складу СУСЗ (рис. 2.6) реалізують наступні функції:

– виконання операцій із вмістом регістрів станів (операцій з ознаками);

– формування сигналу вхідного переносу *CI* для ПЕ і СУП;

– організація арифметичних, логічних і циклічних зсувів слів різної довжини на підставі ознак *Z*, *C*, *N*, *V* результату виконання мікрооперації у АЛБ – реалізує 32 типи зсувів;

– видача одного з шістнадцяти сигналів логічної умови на вихід *CT*, який надходить на БМУ і використовується для організації розгалуження у мікропрограмах.

Розглянемо детально склад та призначення функціональних блоків СУСЗ.

2.3.2. Схема управління

Схема управління (рис.2.6) під дією сигналів мікрокоманди $I_{12} - I_0$ формує внутрішні сигнали, що управляють функціональними блоками СУСЗ.

Для управління СУСЗ в структурі мікрокоманди (рис.2.8) відведено 21 розряд, 13 з яких займає поле інформаційного слова СУСЗ_ІС, що задає операцію виконувану в СУСЗ у поточному такті (рис. 2.10).

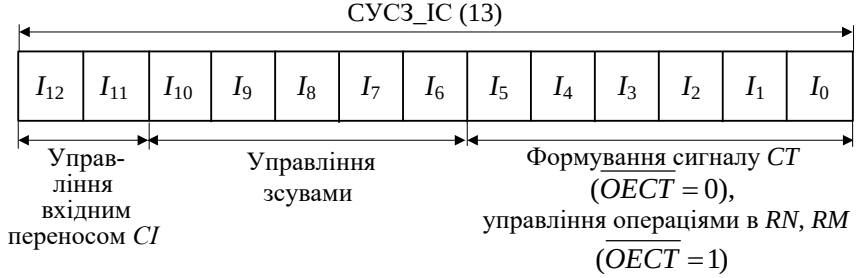


Рис. 2.10. Структура інформаційного поля СУСЗ_ІС мікрокоманди

2.3.3. Блок обробки ознак

Блок обробки ознак (рис. 2.6) складається з двох чотирирозрядних регістрів стану (Рг*N*, Рг*M*), двох вхідних мультиплексорів – *MXN*, *MXM* і одного вихідного мультиплексора – $MX_{\text{ВИХ}}$. Даний блок призначений для зберігання і модифікації ознак, які формуються в ПЕ. Під час виконання мікрооперацій у

АЛБ формуються ознаки – переносу (*C*), знаку результату (*N*), переповнювання (*OVR*) і рівність результату нулю (*Z*), які з виходів *C, N, V, Z* ПЕ поступають на входи *IC*, *IN*, *IV*, *IZ* ознак стану СУСЗ (рис. 2.1, рис. 2.6).

Ознаки зберігаються в регістрах стану РгN і РгM СУСЗ (рис. 2.11). Запис інформації в регістри РгN і РгM відбувається за додатним перепадом тактового імпульсу *T* (перехід від 0 до 1) за наявності сигналу дозволу запису $\overline{CEN} = 0$ *або* $\overline{CEM} = 0$ відповідно.

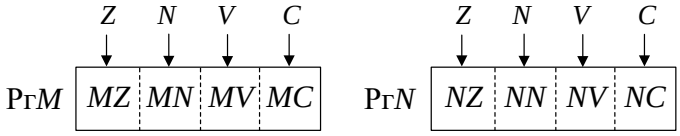


Рис. 2.11. Структура регістрів стану РгN і РгM

2.3.4. Виконання операцій з вмістом регістрів станів

У полі інформаційного слова СУСЗ_ІС якому відповідають розряди $I_5 - I_0$ (рис. 2.10) при встановленні управляючого сигналу $\overline{OECT} = 1$ кодуються операції що можуть бути виконані із вмістом регістрів стану (рис. 2.12).

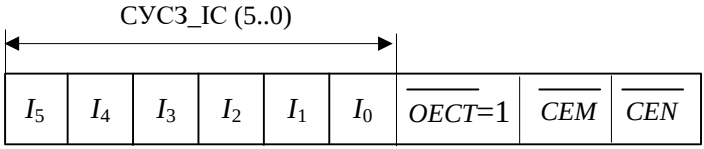


Рис. 2.12. Управління виконанням операцій із вмістом регістрів станів

Операції, що виконуються із вмістом регістрів станів РгN і РгM, можна розділити на наступні групи:

– операції з бітами,
– регістрові операції,
– операції завантаження регістра.

***Операції з вмістом регістру стану* РгN**

В РгN інформація надходить з виходу двоканального мультиплексора *MXN* (рис. 2.6).

Залежно від сигналів мікрокоманди $I_5 - I_0$ в Рг*N* може бути записане слово стану з входів ознак стану СУСЗ – *IC*, *IN*, *IV*, *IZ*, або з виходів регістру Рг*M* – *MC*, *MN*, *MV*, *MZ*. Окрім того, може бути встановлено в значення 0 або 1, як усе інформаційне слово записане у регістрі Рг*N*, так і кожний окремий розряд цього регістру.

Під час запису інформації у регістр Рг*N* на вході дозволу запису встановлюється значення 0 ($\overline{CEN} = 0$). При $\overline{CEN} = 1$ запис в Рг*N* заборонений.

Операції з бітами відповідають встановленню в значення 0 або 1 одного з розрядів Рг*N* в залежності від сигналів $I_5 - I_0$ закодованих у мікрокоманді (табл. 2.9).

Регістрові операції є операціями з інформаційним словом, записаним в Рг*N*. Управління виконанням операцій здійснюється розрядами I_1 і I_0 мікрокоманди за встановленням нульових значень розрядів мікрокоманди $I_5 - I_2$ (табл. 2.10).

Таблиця 2.9. Операції з бітами регістра стану *PrN* ($\overline{CEN} = 0$)

I_5	I_4	I_3	I_2	I_1	I_0	**Операція**	**Коментар**
0	0	1	0	0	1	$1 \to NZ$	Встановлення в 1 ознаки *Z*
0	0	1	0	1	0	$0 \to NC$	Встановлення в 0 ознаки *C*
0	0	1	0	1	1	$1 \to NC$	Встановлення в 1 ознаки *C*
0	0	1	1	0	0	$0 \to NN$	Встановлення в 0 ознаки *N*
0	0	1	1	0	1	$1 \to NN$	Встановлення в 1 ознаки *N*
0	0	1	1	1	0	$0 \to NV$	Встановлення в 0 ознаки *OVR*
0	0	1	1	1	1	$1 \to NV$	Встановлення в 1 ознаки *OVR*

Таблиця 2.10. Регістрові операції з вмістом регістру стану *PrN*, де $i = Z, C, N, V$ ($\overline{CEN} = 0$)

I_5	I_4	I_3	I_2	I_1	I_0	**Операція**	**Коментар**
0	0	0	0	0	0	$M_i \to N_i$	Запис вмісту регістру Рг*M* у регістр Рг*N*
0	0	0	0	0	1	$1 \to N_i$	Встановлення в значення 1 всіх розрядів регістру Рг*N*
0	0	0	0	1	0	$M_i \leftrightarrow N_i$	Регістровий обмін між регістрами Рг*M* та Рг*N*

0 0 0 0 1 1	$0 \to N_i$	Встановлення в значення 0 всіх розрядів регістру Рг*N*

Операції завантаження регістра відповідають запису ознак стану в Рг*N* з входів ознак стану (*IC*, *IN*, *IV*, *IZ*) (табл. 2.11).

Таблиця 2.11. Операції завантаження регістра стану *PrN* ($\overline{CEN} = 0$)

I_5	I_4	I_3	I_2	I_1	I_0	Операція	Коментар
0	0	0	1	1	0	$IZ \to NZ$ $IC \to NC$ $IN \to NN$ $IV \vee NV \to NV$	Використовується під час виконання ряду арифметичних операцій коли немає необхідності перевіряти ознаку переповнювання після кожної операції, а достатньо встановити, що переповнювання було хоча б під час однієї з операцій
0	1	1	0	0	*	$IZ \to NZ$ $\overline{IC} \to NC$ $IN \to NN$ $IV \to NV$	Завантаження з інверсією ознаки переносу
1	0	1	0	0	*		
1	1	1	0	0	*		
0	0	0	1	0	*		
0	1	0	*	*	*	$IZ \to NZ$ $IC \to NC$ $IN \to NN$ $IV \to NV$	Завантаження безпосередньо з входів ознак стану
0	1	1	0	1	*		
0	1	1	1	*	*		
1	0	0	*	*	*		
1	0	1	0	1	*		
1	0	1	1	*	*		
1	1	0	*	*	*		
1	1	1	0	1	*		
1	1	1	1	*	*		

Операції з вмістом регістру стану* Рг*M

В Рг*M* інформація надходить з виходів мультиплексора *MXM* (рис. 2.7). Залежно від значення сигналів $I_5 - I_0$ в Рг*M* може бути записана інформація з входів ознак стану СУСЗ – *IC*, *IN*, *IV*, *IZ* (ціле слово стану, або порозрядно), з виходів регістру Рг*N* – *NC*, *NN*, *NV*, *NZ* або з двоспрямованої шини *Y* – *YC*, *YN*, *YV*, *YZ*. Окрім того в

кожен розряд Рг*M* може бути записане значення 0 або 1. Для виконання запису необхідно, щоб на вході дозволу запису в Рг*M* було встановлене значення $\overline{CEM}=0$. При $\overline{CEM}=1$ запис в Рг*M* заборонений.

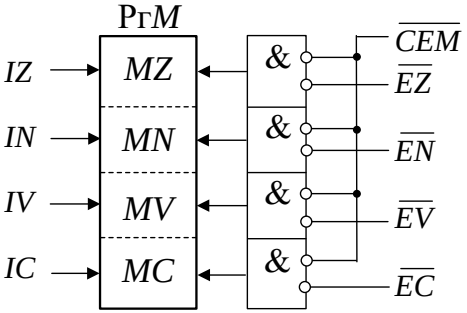


Рис. 2.13. Операції з бітами у регістрі Рг*M*

Операції *запису у біти* регістру Рг*M* ознак результату виконуються з встановленням сигналів дозволу запису $\overline{EZ}=\overline{EN}=\overline{EV}=\overline{EC}=0$ (рис. 2.13). За встановлення управляючого сигналу $\overline{CEM}=0$ і наявністю сигналу дозволу запису у біти регістру відбувається запис ознаки у відповідний розряд регістру Рг*M*. Якщо сигнал на вході дозволу запису ознаки $\overline{CEM}=0$, а значення будь-яких сигналів дозволу запису бітів дорівнює 1, то вміст відповідних розряду регістру Рг*M* не зміниться. Тобто, під час встановлення сигналів $\overline{EZ}\vee\overline{EN}\vee\overline{EV}\vee\overline{EC}=1$ відбувається заборона запису у відповідний розряд регістру Рг*M*. За значення $\overline{CEM}=1$ забороняється запис у всі розряди регістру Рг*M*.

Управління виконанням регістрових операції і операції завантаження, що виконуються у регістрі Рг*M*, наведено в табл. 2.12 – 2.13.

Таблиця 2.12. Регістрові операції регістра Рг*M*
($\overline{EZ}=\overline{EN}=\overline{EV}=\overline{EC}=\overline{CEM}=0$)

I_5	I_4	I_3	I_2	I_1	I_0	Операція	Коментар
0	0	0	0	0	0	$Y_i \to M_i$	Запис інформації з шини *Y* в Рг*M*
0	0	0	0	0	1	$1 \to M_i$	Встановлення в 1 всіх розрядів Рг*M*

0	0	0	0	1	0	$N_i \leftrightarrow M_i$	Регістровий обмін
0	0	0	0	1	1	$0 \rightarrow M_i$	Встановлення в 0 всіх розрядів РгM
0	0	0	1	0	1	$\overline{M_i} \rightarrow M_i$	Інвертування вмісту регістру РгM

Таблиця 2.13. Операції завантаження регістру РгM ($\overline{EZ} = \overline{EN} = \overline{EV} = \overline{EC} = \overline{CEM} = 0$)

I_5	I_4	I_3	I_2	I_1	I_0	Операція	Коментар
0	0	0	1	0	0	$IZ \rightarrow NZ$ $IV \rightarrow MV$ $IN \rightarrow MN$ $IC \rightarrow MC$	Використовується при організації зсувів з використанням ознаки переповнювання, а не переносу
0	0	1	0	0	*	$IZ \rightarrow MZ$	Завантаження з інверсією ознаки переносу
0	1	1	0	0	*	$\overline{IC} \rightarrow MC$	
1	0	1	0	0	*	$IM \rightarrow MN$	
1	1	1	0	0	*	$IV \rightarrow MV$	
0	0	0	1	1	*	$IZ \rightarrow MZ$	Завантаження безпосередньо з входів ознак стану
0	0	1	0	1	*	$IC \rightarrow MC$	
0	0	1	1	*	*	$IN \rightarrow MN$	
0	0	1	1	*	*	$IV \rightarrow MV$	
0	1	0	*	*	*		
0	1	1	0	1	*		
0	1	1	1	*	*		
1	0	0	*	*	*		
1	0	1	0	1	*		
1	0	1	1	*	*		
1	1	0	*	*	*		
1	1	1	*	*	*		

Приклад 2.2. Розробити мікропрограму для БОД, що реалізує заданий мікроалгоритм (рис. 2.14).

Виконання завдання

У заданому мікроалгоритмі на рис. 2.14 мнемоніка *LOAD* визначає операції завантаження регістрів стану (*RN, RM*) або відповідних бітів цих регістрів (*MC, MZ* та інших). У табл. 2.14

наведена кодова карта мікропрограми. У структурі закодованих мікрокоманд умовно показані тільки ті поля, що управляють АЛБ та СУСЗ. В табл. 2.15 наведена діаграма вмісту регістру стану *RM* на протязі виконання мікропрограми.

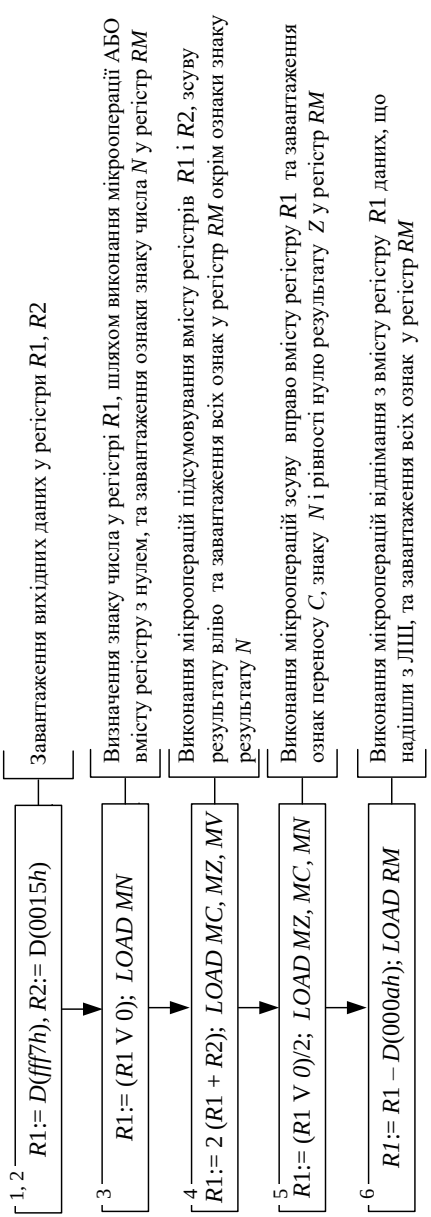


Рис. 2.14. Вихідний мікроалгоритм

Таблиця 2.14. Кодова карта

№ такту	Адреса [16]	Константа БОД		БОД													
				ВС1 (АЛБ)				ВР2 (СУСЗ)									
		D [16]	$\overline{OED}$	АЛБ_МІ [2]	A [16]	B [16]	$\overline{OEY}$	СУСЗ_МІ [2]	$\overline{EC}$	$\overline{EZ}$	$\overline{EN}$	$\overline{EV}$	$\overline{CEN}$	$\overline{CEM}$	$\overline{OECT}$	$\overline{SE}$	
1	0001	*FFF7*	0	011 011 101	1	1	1	00 00000 000000	0	0	0	0	1	1	1	1	
2	0002	0015	0	011 011 101	2	2	1	00 00000 000000	0	0	0	0	1	1	1	1	
3	0003	****	1	011 011 100	1	1	1	00 00000 000110	1	1	0	1	1	0	1	1	
4	0004	****	1	111 000 001	2	1	1	00 10000 000110	0	0	1	0	1	0	1	0	
5	0005	****	1	101 011 100	1	1	1	00 00010 000110	0	0	0	1	1	0	1	0	
6	0006	001*A*	0	011 001 101	1	1	1	01 00000 000110	0	0	0	0	1	0	1	1	

Таблиця 2.15. Діаграма вмісту регістру стану *RM*

№ такту	***MC***	***MZ***	***MN***	***MV***
3	*	*	1	*
4	1	0	*	0
5	0	0	*	0
6	0	0	1	0

Результат налагодження мікропрограми у моделюючій програмі *COMPLEX* представлений на рис. 2.15.

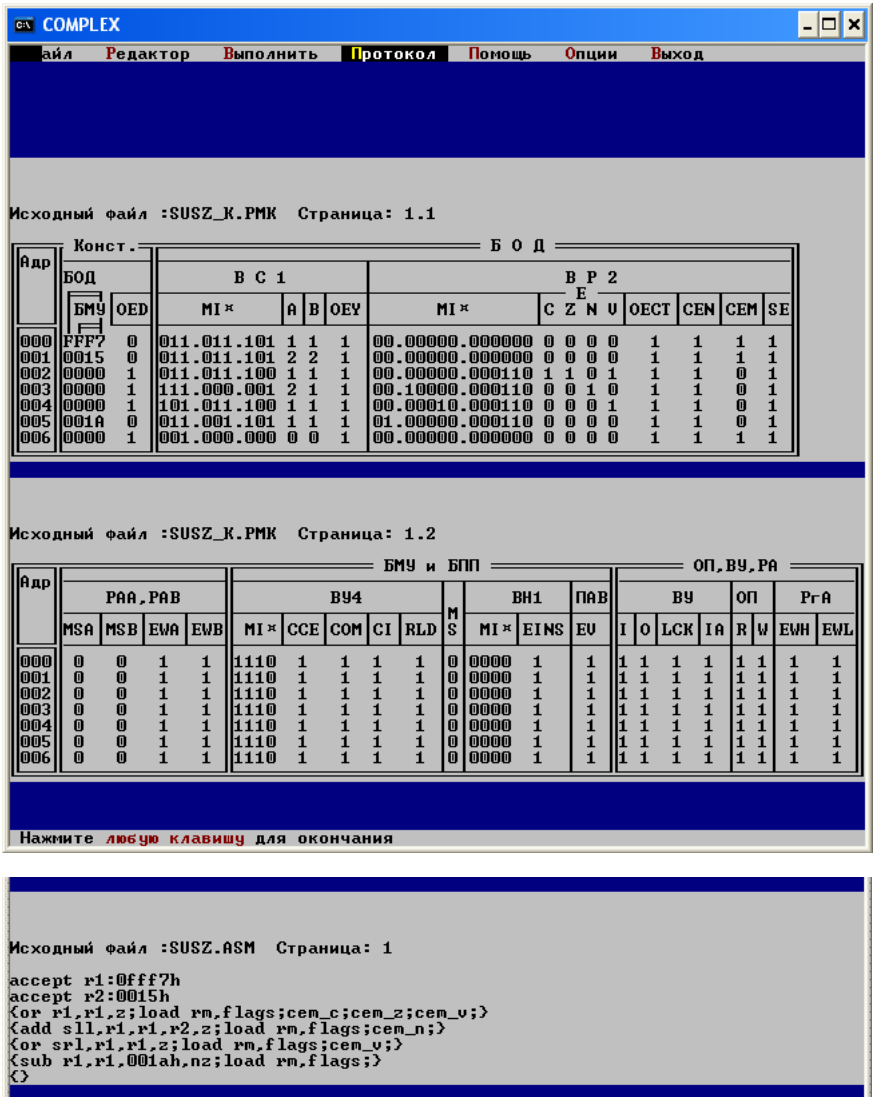


Рис. 2.15. Мікропрограма управління станами

Вивід вмісту регістрів* Рг*N*, Рг*M* *на двоспрямовану шину* *Y

Інформація з виходів Рг*N*, Рг*M* або з входів ознак стану СУСЗ – *IC*, *IN*, *IV*, IZ, через вихідний мультиплексор із тристабільними виходами $МХ_{Вих}$ (рис. 2.6) надходить на двоспрямовану шину *Y*. Якщо на входах $I_5 - I_0$ встановлені нульові значення, то шина *Y* є вхідною незалежно від сигналу дозволу виводу інформації ($\overline{OEY}$). В інших випадках шина *Y* є вихідною. Управління виводом інформації через шину *Y* за допомогою сигналів $\overline{OEY}$, I_5 та I_4 надане в табл. 2.16.

Таблиця 2.16. Управління виводом інформації через шину *Y* ($i = Z, C, N, V$)

$\overline{OEY}$	I_5	I_4	*Y*	Коментар
1	*	*	*Z*	Стан високого опору
0	0	*	$N_i \to Y_i$	Вивід вмісту Рг*N*
0	1	0	$M_i \to Y_i$	Вивід вмісту Рг*M*
0	1	1	$I_i \to Y_i$	Вивід інформації з входів ознак стану
*	$(I_5..I_0) = 000000$ $Y_i \to$ Рг*M*			При значенні 0 на входах $I_5 - I_0$ шина *Y* є вхідною незалежно від значення сигналу $\overline{OEY}$

2.3.5. Блок перевірки умови

Блок перевірки умови (рис. 2.6) складається з комбінаційної схеми перевірки умови та мультиплексора зі схемою управління полярністю.

Блок перевірки умови призначений для формування вихідного сигналу коду умови *CT*, яка надходить на один з входів мультиплексора умов МУ (рис. 2.7) БМУ і використовується для організації різноманітних переходів у мікропрограмах.

Управління формуванням умови здійснюється розрядами $I_5 - I_0$ поля СУСЗ_ІС МК (рис. 2.10) за встановлення сигналу $\overline{OECT} = 0$ (рис. 2.16).

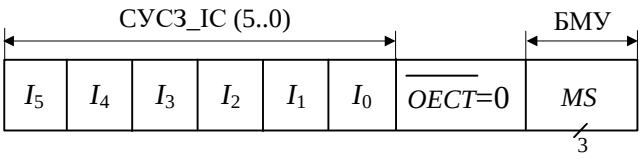


Рис. 2.16. Управління формуванням сигналу *CT*

Блок перевірки умови виконує одну з 16 можливих операцій (табл. 2.17) формування умови, результат якої видається на вихід коду умови *CT*. Вибір операндів для виконання операцій в даному блоці управляється сигналами мікрокоманди I_4, I_5.

Найбільш поширеними операціями, що виконуються блоком перевірки умови є передавання однієї з ознак стану на вихід *CT* (чотири операції). Ще чотири операції є більш складними і застосовуються під час виконання в АЛБ операції віднімання ($A-B$) для забезпечення умов $A=B$, $A\neq B$, $A\geq B$ та інших. (табл. 2.18). При цьому аргументи *A* і *B* мають бути подані в доповнювальному коді або як числа без знаку.

Далі результат однієї з виконаної у блоці операції формування умови через мультиплексор передається на вхід ***схеми управління полярністю***, яка за необхідності інвертує отриманий результат. Таким чином решта вісім операцій є інверсією вже розглянутих операцій. Результат перевірки умови з виходу схеми управління полярністю *CO* надходить на тристабільну шину *CT*, що управляється сигналом дозволу коду умови $\overline{OECT}$.

Для підключення виходу *CT* СУСЗ до одного з входів мультиплексора МУ БМУ в трирозрядному полі *MS* мікрокоманди, що належить до полів управління БМУ необхідно встановити номер входу мультиплексора (рис. 2.16).

При $\overline{OECT}=0$ дозволяється вивід коду умови через шину *CT*. Якщо $\overline{OECT}=1$, то вивід коду умови заборонений, а шина *CT* знаходиться в стані високого опору.

Таблиця 2.17. Управління виходом коду умови *CT* ($\overline{OECT} = 0$)

I_3 I_2 I_1 I_0	$I_5 = I_4 = 0$	$I_5 = 0, I_4 = 1$	$I_5 = 1, I_4 = 0$	$I_5 = I_4 = 1$
0 0 0 0	$(NN \oplus NV) \vee NZ$	$(NN \oplus NV) \vee NZ$	$(MN \oplus MV) \vee MZ$	$(IN \oplus IV) \vee IZ$
0 0 0 1	$\overline{(NN \oplus NV)} \,\&\, \overline{NZ}$	$\overline{(NN \oplus NV)} \,\&\, \overline{NZ}$	$\overline{(MN \oplus MV)} \,\&\, \overline{MZ}$	$\overline{(IN \oplus IV)} \,\&\, \overline{IZ}$
0 0 1 0	$NN \oplus NV$	$NN \oplus NV$	$MN \oplus MV$	$IN \oplus IV$
0 0 1 1	$\overline{NN \oplus NV}$	$\overline{NN \oplus NV}$	$\overline{MN \oplus MV}$	$\overline{IN \oplus IV}$
0 1 0 0	NZ	NZ	MZ	IZ
0 1 0 1	$\overline{NZ}$	$\overline{NZ}$	$\overline{MZ}$	$\overline{IZ}$
0 1 1 0	NV	NV	MV	IV
0 1 1 1	$\overline{NV}$	$\overline{NV}$	$\overline{MV}$	$\overline{IV}$
1 0 0 0	$NC \vee NZ$	$NC \vee NZ$	$MC \vee MZ$	$IC \vee IZ$
1 0 0 1	$\overline{NC} \,\&\, \overline{NZ}$	$\overline{NC} \,\&\, \overline{NZ}$	$\overline{MC} \,\&\, \overline{MZ}$	$\overline{IC} \,\&\, \overline{IZ}$
1 0 1 0	NC	NC	MC	IC
1 0 1 1	$\overline{NC}$	$\overline{NC}$	$\overline{MC}$	$\overline{IC}$
1 1 0 0	$\overline{NC} \vee NZ$	$\overline{NC} \vee NZ$	$\overline{MC} \vee MZ$	$\overline{IC} \vee IZ$
1 1 0 1	$NC \,\&\, \overline{NZ}$	$NC \,\&\, \overline{NZ}$	$MC \,\&\, \overline{MZ}$	$IC \,\&\, \overline{IZ}$
1 1 1 0	$IN \oplus MN$	NN	MN	IN
1 1 1 1	$\overline{IN \oplus MN}$	$\overline{NN}$	$\overline{MN}$	$\overline{IN}$

Таблиця 2.18. Перевірка відношення чисел A і B після виконання операції (A – B)

Відношення	**Числа без знаку**								
	Стан	***CT* = 1**				***CT* = 0**			
		I_3	I_2	I_1	I_0	I_3	I_2	I_1	I_0
$A = B$	$Z = 1$	0	1	0	0	0	1	0	1
$A \neq B$	$Z = 0$	0	1	0	1	0	1	0	0
$A \geq B$	$C = 1$	1	0	1	0	1	0	1	1
$A < B$	$C = 0$	1	0	1	1	1	0	1	0
$A > B$	$C \,\&\, Z = 1$	1	1	0	1	1	1	0	0
$A \leq B$	$C \vee Z = 1$	1	1	0	0	1	1	0	1
Відношення	**Числа в доповнювальному коді**								
	Стан	***CT* = 1**				***CT* = 0**			
		I_3	I_2	I_1	I_0	I_3	I_2	I_1	I_0
$A = B$	$Z = 1$	0	1	0	0	0	1	0	1
$A \neq B$	$Z = 0$	0	1	0	1	0	1	0	0
$A \geq B$	$\overline{N \oplus V} = 1$	0	0	1	1	0	0	1	0
$A < B$	$N \oplus V = 1$	0	0	1	0	0	0	1	1
$A > B$	$[\overline{N \oplus V}] \,\&\, \overline{Z} = 1$	0	0	0	1	0	0	0	0
$A \leq B$	$[N \oplus V] \,\&\, Z = 1$	0	0	0	0	0	0	0	1

2.3.6. Блок управління переносами

Блок управління переносами (рис. 2.6) формує сигнал вихідного переносу *CO*, який використовується для формування вхідного переносу *CI* АЛБ процесорного елементу. Управління формуванням вихідного переносу здійснюється сигналами $I_{12}, I_{11}, I_5, I_3, I_2, I_1$ поля СУЗС_ІС мікрокоманди як показано в табл. 2.19.

Розряди I_{12}, I_{11} використовуються для присвоєння вхідному переносу значення 0 або 1. Якщо $I_{12}I_{11} = 00$, то $CI = 0$, та якщо $I_{12}I_{11} = 01$, $CI = 1$.

Якщо для формування вихідного переносу окрім розрядів I_{12}, I_{11} (коли $I_{12}I_{11} = 11$) використовують також розряди

I_5, I_3, I_2, I_1 поля СУСЗ_ІС МК то вхідному переносу *CI* можна присвоїти значення розрядів переносу *MC, NC* регістрів Рг*M* та Рг*N,* або їх заперечення.

Таблиця 2.19. Управління формуванням сигналу вхідного переносу *CI* АЛБ

I_{12}	I_{11}	I_5	I_3	I_2	I_1	***CO***	**Коментар**
0	0	*	*	*	*	0	
0	1	*	*	*	*	1	
1	0	*	*	*	*	*CX*	
1	1	0	*	*	*	*NC*	$I_3I_2I_1 \neq 100$
		1	*	*	*	*MC*	$I_3I_2I_1 \neq 100$
		0	1	0	0	$\overline{NC}$	
		1	1	0	0	$\overline{MC}$	

Таким чином, в якості вхідного переносу *CI* використовується одне з наступних джерел – 0, 1, *CX*, *NC*, *MC*, $\overline{NC}$, $\overline{MC}$, що дозволяє легко реалізувати операції додавання та віднімання чисел як звичайної так і подвійної довжини. Вхід *CX* (коли $I_{12}I_{11} = 10$) служить для організації виконання ПЕ К1804ВС2 ряду спеціальних функцій при поєднанні входу *CX* з виходом *Z* ПЕ.

2.3.7. Блок управління зсувами

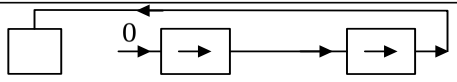
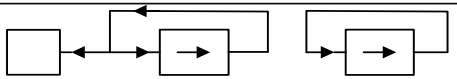
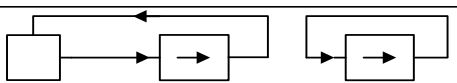
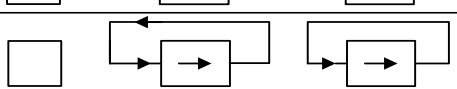
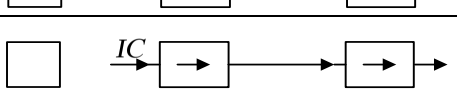
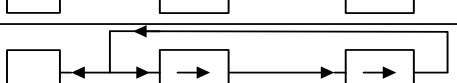
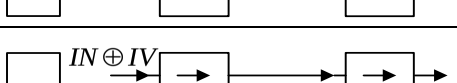
Блок управління зсувами (рис. 2.6) призначений для організації різних варіантів арифметичних, логічних та циклічних зсувів. Для управління завданням типу зсувів використовуються сигнали $I_{10} - I_6$ поля СУСЗ_ІС мікрокоманди. Загалом можливі 32 типи зсувів – 16 вліво і 16 вправо (табл. 2.20).

Виводи зсуву *PF*0, *PF*3, *PQ*0, *PQ*3 є тристабільними і управляються сигналом $\overline{SE}$. Під час встановлення $\overline{SE} = 0$ зсув дозволений, інакше $\overline{SE} = 1$ зсув заборонений, а всі виводи зсуву знаходяться в стані високого опору.

Таблиця 2.20. Управління зсувами ($\overline{SE} = 0$)

Розряди МК				Модифікація зсуву			Сигнали на виводах ВР2				
1				2			3				
I_{10}	I_9	I_8	I_7	I_6	*MC*	ЗС*B*	ЗС*Q*	*PF*0	*PF*3	*PQ*0	*PQ*3	*MC*
0	0	0	0	0		0	0	Z	0	Z	0	–
0	0	0	0	1		1	1	Z	1	Z	1	–
0	0	0	1	0		0	*MN*	Z	0	Z	*MN*	PF0
0	0	0	1	1		1		Z	1	Z	PF0	–
0	0	1	0	0				Z	*MC*	Z	PF0	–
0	0	1	0	1		*MN*		Z	*MN*	Z	PF0	–
0	0	1	1	0		0		Z	*0*	Z	PF0	–

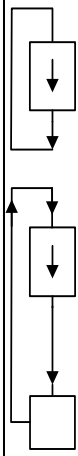
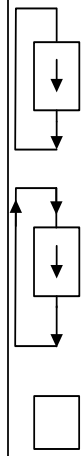
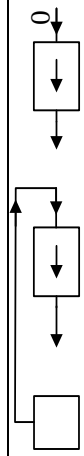
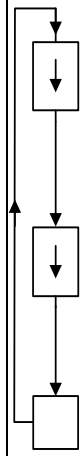
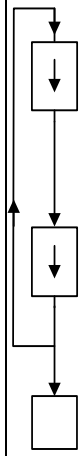
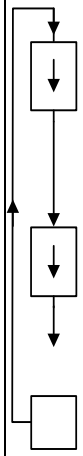
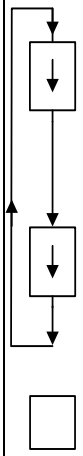
Продовження таблиці 2.16.

1					2	3				
0	0	1	1	1		*Z*	*0*	*Z*	P*F*0	P*Q*0
0	1	0	0	0		*Z*	P*F*0	*Z*	P*Q*0	P*F*0
0	1	0	0	1		*Z*	*MC*	*Z*	P*Q*0	P*F*0
0	1	0	1	0		*Z*	P*F*0	*Z*	P*Q*0	–
0	1	0	1	1		*Z*	*IC*	*Z*	P*F*0	–
0	1	1	0	0		*Z*	*MC*	*Z*	P*F*0	P*Q*0
0	1	1	0	1		*Z*	P*Q*0	*Z*	P*F*0	P*Q*0
0	1	1	1	0		*Z*	*IN ⊕ IV*	*Z*	P*F*0	–
0	1	1	1	1		*Z*	P*Q*0	*Z*	P*F*0	–

Продовження таблиці 2.16.

1	0	0	0	0		0	Z	0	Z	PF3
1	0	0	0	1		1	Z	1	Z	PF3
1	0	0	1	0		0	Z	0	Z	–
1	0	0	1	1		1	Z	1	Z	–
1	0	1	0	0		PQ3	Z	0	Z	PF3
1	0	1	0	1		PQ3	Z	1	Z	PF3
1	0	1	1	0		PQ3	Z	0	Z	–
1	0	1	1	1		PQ3	Z	1	Z	–
1	1	0	0	0		PF3	Z	PQ3	Z	PF3

Продовження табл. 2.16

1	1	0	0	1		*MC*	Z	PQ3	Z	PF3
1	1	0	1	0		PF3	Z	PQ3	Z	–
1	1	0	1	1		*MC*	Z	0	Z	–
1	1	1	0	0		PQ3	Z	*MC*	Z	PF3
1	1	1	0	1		PQ3	Z	PF3	Z	PF3
1	1	1	1	0		PQ3	Z	*MC*	Z	–
1	1	1	1	1		PQ3		PF3		–

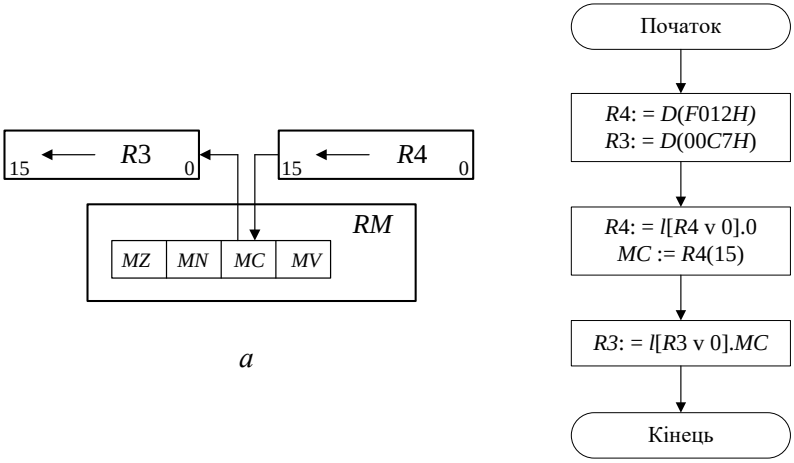
Приклад 2.3. Розробити фрагмент мікропрограми для зсуву вліво слова подвійної довжини. Старша і молодша частини слова записані у регістрах *R*3, *R*4 відповідно.

Виконання завдання:

Операційна схема та мікроалгоритм виконання операції наведена на рис. 2.17, *а*, *б*. Для реалізації зсуву слова подвійної довжини виконаємо два типи зсувів: по-перше – логічний зсув *SL*.16 ($I_{10}..I_6 = 10000$) молодшої частини слова *R*4, під час чого старший розряд, що виходить за межі розрядної сітки, зберігається у розряді *MC* регістру стану *RM*. Далі виконаємо циклічний зсув *SL*.25 ($I_{10}..I_6 = 11001$) старшої частини слова *R*3, під час чого у молодший розряд, звільнений за зсуву, записується вміст розряду *MC* регістру *RM*, в якому в подальшому зберігається старший розряд слова, що вийшов за межі розрядної сітки.

Цифрова діаграма стану регістрів наведена у табл. 2.21.Кодова карта розробленої мікропрограми наведена у табл. 2.22.

Мікропрограма зсуву слів подвійної довжини, налагоджена за допомогою середи модулювання *COMPLEX*, наведена на рис. 2.18.



а *б*

Рис. 2.17. Реалізація зсуву вліво слова подвійної довжини: а – операційна схема; б – мікроалгоритм.

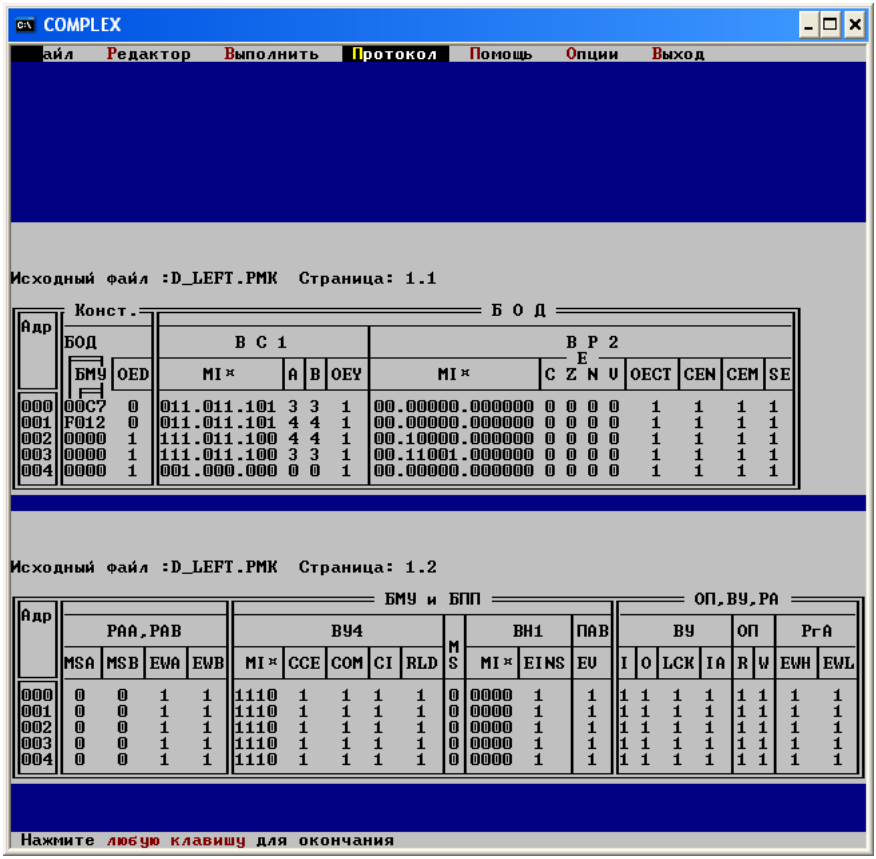
Таблиця 2.21. Цифрова діаграма стану регістрів

№ такту	Вихідний стан			Інформація у приймачі результату, [16]	МС	Мікрооперація
	Шина *D*, [16]	*R*3, [16]	*R*4, [16]			
1	00*C*7	0000	0000	$R3 := 00C7$	*	$R3 := D(007C) \vee 0$
2	*F*012	00*C*7	0000	$R4 := F012$	*	$R4 := D(F012) \vee 0$
3	****	00*C*7	*F*012	$R4 := E024$	1	$R4 := SLL(R4 \vee 0)$
4	****	0070	*E*024	$R3 := 018F$	1	$R3 := S.25(R3 \vee 0)$
		018*F*	*E*024			Результат

Примітка: *SLL, S*.25 – мнемонічне позначення зсувів логічного вправо та циклічного відповідно.

Таблиця 2.22 Кодова карта мікропрограми зсуву слів подвійної довжини

№ такту	Адреса [16]	Константа БОД		БОД													
				ВС1 (АЛБ)				ВР2 (СУСЗ)									
		D [16]	$\overline{OED}$	АЛБ_МІ [2]	A [16]	B [16]	$\overline{OEY}$	СУСЗ_МІ [2]	$\overline{EC}$	$\overline{EZ}$	$\overline{EN}$	$\overline{EV}$	$\overline{CEN}$	$\overline{CEM}$	$\overline{OECT}$	$\overline{SE}$	
1	0000	00*C*7	0	011 011 101	3	3	1	00 00000 000000	0	0	0	0	1	1	1	1	
2	0001	*F*012	0	011 011 101	4	4	1	00 00000 000000	0	0	0	0	1	1	1	1	
3	0002	****	1	111 011 100	4	4	1	00 10000 000000	0	0	0	0	1	1	1	0	
4	0003	****	1	111 011 100	3	3	1	00 11001 000000	0	0	0	0	1	1	1	0	



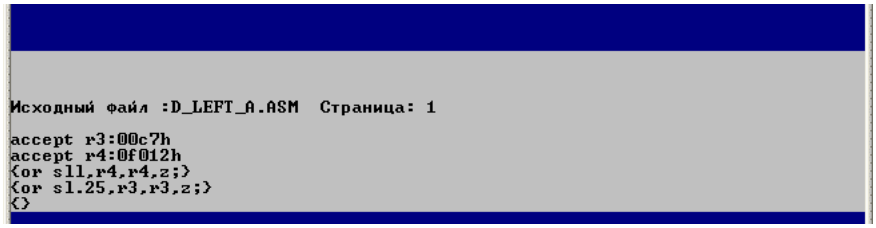
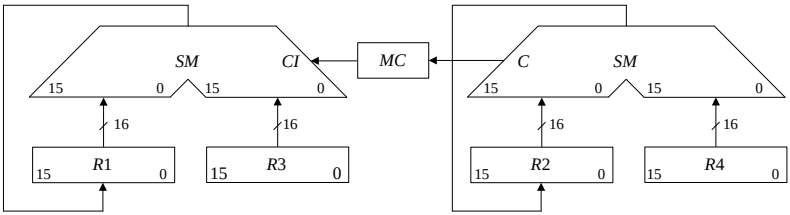


Рис. 2.18.Мікропрограма зсуву вліво слова подвійної довжини

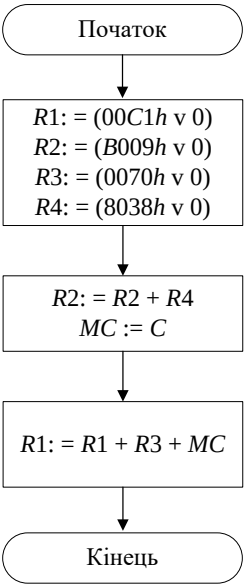
Приклад 2.4. Розробити фрагмент мікропрограми для додавання слів подвійної довжини. Доданки записані у регістрах *R*1, *R*2 та *R*3, *R*4 відповідно.

Виконання завдання:

Операційна схема та мікроалгоритм виконання операції наведена на рис. 2.19, *а*, *б*.



а



б

Рис. 2.19. Операція додавання слів подвійної довжини: *а* – операційна схема; *б* – мікроалгоритм

Для реалізації додавання слів подвійної довжини виконаємо послідовне додавання молодших і старших частин слова із поширенням переносу із молодшої частини результату у старшу. Для реалізації поширення переносу під час додавання молодших частин слів збережемо ознаку переносу у розряді *MC* регістру стану *RM*:

$$R2 := R2 + R4 \,;\, LOAD\ MC,$$

під час додавання старших частин, подамо значення переносу на вхід *CI* АЛБ:

$$R1 := R1 + R3 + CI, \text{ де } CI := MC.$$

Цифрова діаграма стану регістрів наведена у табл. 2.23.Кодова карта розробленої мікропрограми наведена у табл. 2.24.

Мікропрограма додавання слів подвійної довжини налагоджена за допомогою середи модулювання *COMPLEX* наведена на рис. 2.20.

2.3.8. Виконання операції нормалізації

ІС К1804ВР2 забезпечує виконання операції нормалізації чисел звичайної і подвійної довжини, як для ПЕ К1804ВС1, так і для ПЕ К1804ВС2.

Поєднання виводів мікросхем К1804ВР2, К1804ВС1 і К1804ВУ4 показано на рис. 2.7.

В мікросхемі К1804ВС1 однією з ознак закінчення нормалізації під час виконання чергової мікрооперації служить встановлення сигналу на виході *CO* – старшої секції результату (під час реалізації обчислень у МПС із словами довжиною більш ніж чотири розряди ПЕ поєднуються у ланцюги, див розділ 2.4 рис. 2.22). При цьому на виході *CO* формується сигнал, як результат виконання операції ВИКЛЮЧНЕ АБО над двома старшими розрядами.

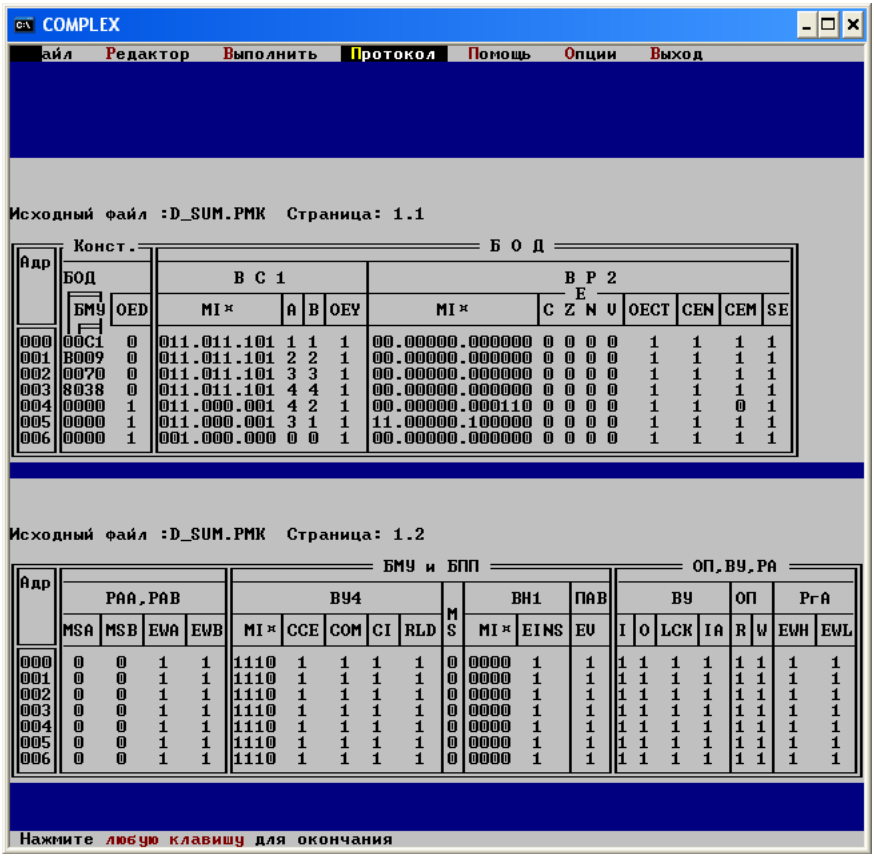
Другою ознакою, що інформує про закінчення нормалізації служить сигнал на виході *OVR*, що є результатом операції ВИКЛЮЧНЕ АБО над двома середніми розрядами старшої секції результату. Ця ознака встановлюється за один такт до закінчення операції. Сигнали *CO* або *OVR* через блок перевірки умови СУСЗ виставляються на виході *CT* К1804ВС2 та надходять на вхід умови *CC* БМУ ІС К1804ВУ4 (див. розділ 2.5, рис. 2.27).

Таблиця 2.23. Цифрова діаграма стану регістрів

№ такту	Вихідний стан					Інформація у приймачі результату, [16]	МС	Мікрооперація
	Шина ***D***, [16]	***R*1**, [16]	***R*2**, [16]	***R*3**, [16]	***R*4**, [16]			
1	00*C*1	0000	0000	0000	0000	$R1 := 00C1$	*	$R1 := D(00C1) \vee 0$
2	0009	00*C*1	0000	0000	0000	$R2 := 0009$	*	$R2 := D(0009) \vee 0$
3	0070	00*C*1	0009	0000	0000	$R3 := 0070$	*	$R3 := D(0070) \vee 0$
4	8038	00*C*1	0009	0070	0000	$R4 := 8038$	*	$R4 := D(8038) \vee 0$
4	****	00*C*1	0009	0070	8038	$R2 := 3041$	1	$R2 := R2 + R4 + CI$
5	****	00*C*1	3041	0070	8038	$R1 := 00CB$	*	$R1 := R3 + R1 + 1(MC)$
		00*CB*	3041					Результат

Таблиця 2.24 Кодова карта мікропрограми додавання слів подвійної довжини

№ такту	**Адреса** [16]	**Константа БОД**		**БОД**													
				ВС1 (АЛБ)				**ВР2 (СУСЗ)**									
		D [16]	$\overline{OED}$	АЛБ_МІ [2]	A [16]	B [16]	$\overline{OEY}$	СУСЗ_МІ [2]	$\overline{EC}$	$\overline{EZ}$	$\overline{EN}$	$\overline{EV}$	$\overline{CEN}$	$\overline{CEM}$	$\overline{OECT}$	$\overline{SE}$	
1	0000	00*C*1	0	011 011 101	1	1	1	00 00000 000000	0	0	0	0	1	1	1	1	
2	0001	0009	0	011 011 101	2	2	1	00 00000 000000	0	0	0	0	1	1	1	1	
3	0002	0070	0	011 011 101	3	3	1	00 00000 000000	0	0	0	0	1	1	1	1	
4	0003	8038	0	011 011 101	4	4	1	00 00000 000000	0	0	0	0	1	1	1	1	
5	0004	****	1	011 000 001	4	2	1	00 00000 000110	0	0	0	0	1	0	1	1	
6	0005	****	1	011 000 001	3	1	1	11 00000 100000	0	0	0	0	1	1	1	1	



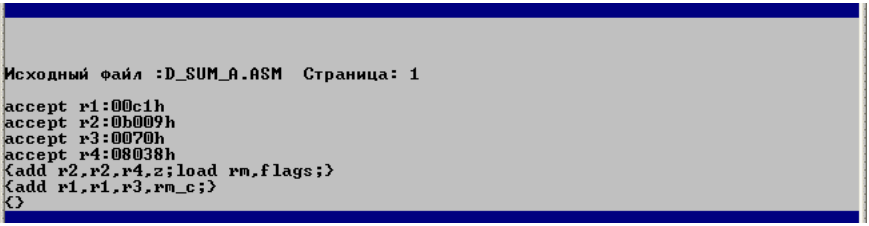


Рис. 2.20. Мікропрограма додавання слів подвійної довжини

Оскільки розглянуті ознаки записуються в регістр стану з запізненням на один такт, то при використанні в якості ознаки закінчення нормалізації сигналу сформованого на виході *C*4 необхідно останнім кроком нормалізації виконати зсув вбік молодших розрядів. Такий зсув виконується СУСЗ спеціальною операцією зсуву за надходження на входи $I_{10} - I_6$ відповідних кодів: $I_{10}, I_9, I_8, I_7, I_6 = 01001$ – нормалізація чисел подвійної довжини або $I_{10}, I_9, I_8, I_7, I_6 = 00010$ – нормалізація чисел звичайної довжини.

В процесорному елементі К1804ВС1 не передбачена спеціальна мікрооперація нормалізації. Нормалізація виконується з застосуванням мікрооперації зсуву. Для виконання функції елемента ВИКЛЮЧНЕ АБО, який відсутній в мікросхемі К1804ВС1, в СУСЗ передбачена спеціальна операція ($MN \oplus IN$), що виконується із знаковими розрядами числа сформованими у попередньому такті обчислень (*MN*) і у поточному такті (*IN*).

2.3.9. Реалізація переривань

Мікросхема К1804ВР2 може використовуватися для організації переривань як на програмному, так і на мікропрограмному рівні. При виконанні переривань відбувається передавання ознак стану з регістру Рг*M* (під час переривань програмного рівня) або з регістрів Рг*N*, Рг*M* (під час переривань мікропрограмного рівня) в зовнішні регістри. Таке передавання ознак стану реалізується через двоспрямовану шину *Y* (див. табл. 2.16).

Після закінчення переривання в Рг*M* і Рг*N* повинен бути відновлений стан, що передував перериванню. Вміст Рг*N*, що передував перериванню, спочатку записується в Рг*M*, а далі за запису в Рг*M* ознак стану, що передували перериванню, вміст Рг*M* передається в Рг*N*.

Організація системи переривань в мікропроцесорних системах побудованих на секціонованих інтегральних схемах детально розглянута у розділі 2.6. Загальні принципи організації та функціонування системи переривань у мікропроцесорних системах розглянуті в розділі 5.

2.4. Блоки обробки даних

Блоки обробки даних будуються на основі процесорних елементів, наприклад, ІС К1804ВС1. Нарощування довжини розрядної сітки процесора забезпечується поєднанням декількох ПЕ, як показано на рис. 2.21. для випадку, коли $n = 16$. Шини *AA, AB, I* і *CLK*, які умовно не зображені, підключаються паралельно до кожної ІС ВС1, аналогічно шині *OE*. З ціллю зменшення часу розповсюдження переносу використовують ІС К1804ВР1, умовне графічне позначення якої показано на рис. 2.22. В цьому випадку (коли $n = 16$) ланцюги розповсюдження переносу реалізують, відповідно схемі зображеній на рис. 2.23.

Мікросхеми ВС1 мають відповідні виходи *P* і G, що дозволяє при побудові процесорів з довжиною слова $n > 16$ здійснювати каскадне підключення цих мікросхем (рис. 2.23).

Часові параметри ІС К1804ВР1 приведені в табл. 2.25.

Таблиця 2.25. Параметри ІС К1804ВР1.

Час затримки сигналів	**Значення, нс**
від *CI* до *CO*	10
від *P*0 – *P*3 до *P*	8,5
від *P*0 – *P*3, *G*0 – *G*3 до *CX, CY, CZ*	7
від *P*0 – *P*3, *G*0 – *G*3 до *G*	9

Виходи ознак і зсувачів підключають до однойменних входів ІС К1804ВР2 (рис. 2.9). Відповідні виходи цієї ІС забезпечують подачу на блок управління різних ознак і вхідного переносу для молодшої секції процесору.

Для управління роботою шістнадцятирозрядного операційного блоку у структурі мікрокоманди задіяні наступні поля (рис. 2.8):

що управляють ІС К1804ВС1 – АЛБ_МІ, *AB, AA,* $\overline{OE}$;

що управляють ІС К1804ВР2 – СУСЗ_МІ, $\overline{OECT}$, $\overline{EZ}$, $\overline{EN}$, $\overline{EC}$, $\overline{EV}$, $\overline{CEN}$, $\overline{CEM}$, $\overline{OEY}$, $\overline{SE}$.

Дані приймаються в операційний блок по шині *D*, а видаються по шині *Y* ІС К1804ВС1. Двоспрямована шина *Y* ІС К1804ВР2 дозволяє, наприклад, зберегти в стеку і завантажити ознаки при роботі з підпрограмами.

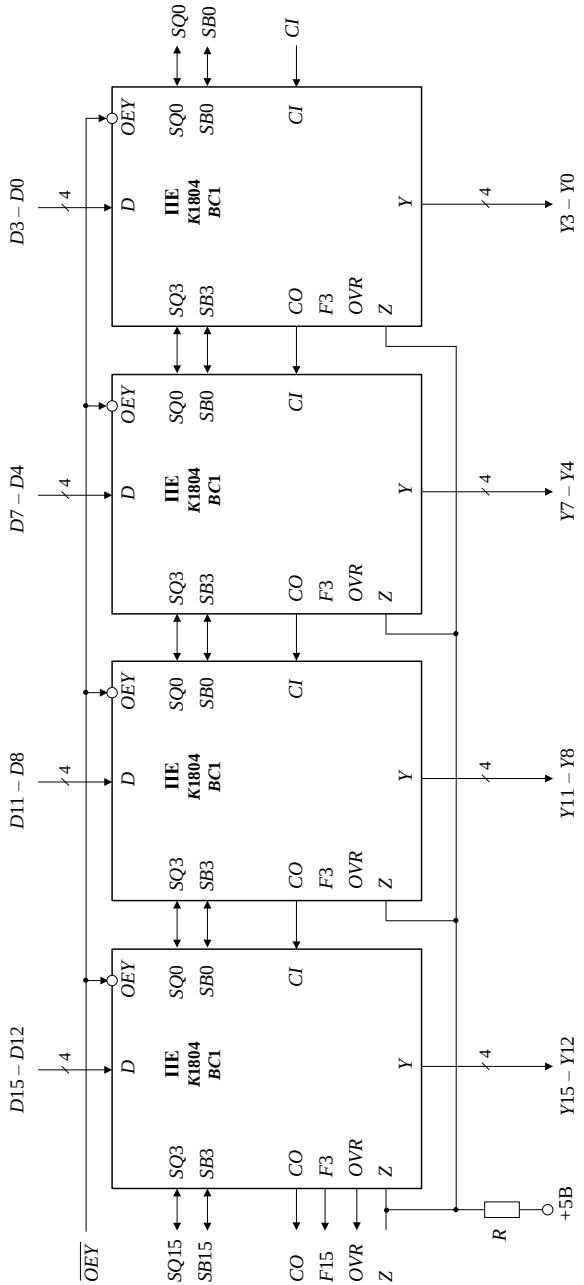


Рис 2.21. Структурна схема шістнадцятирозрядного ПЕ

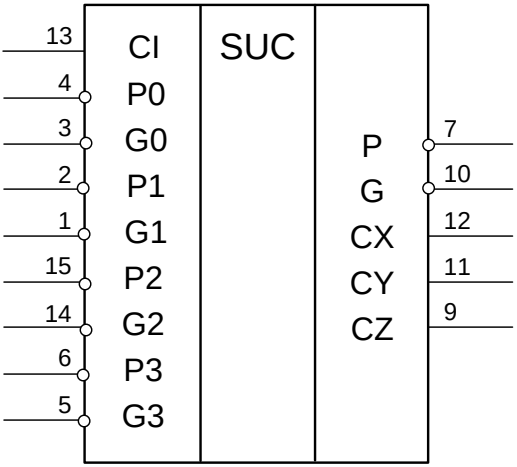


Рис. 2.22 Умовне графічне позначення ІС К1804ВР1

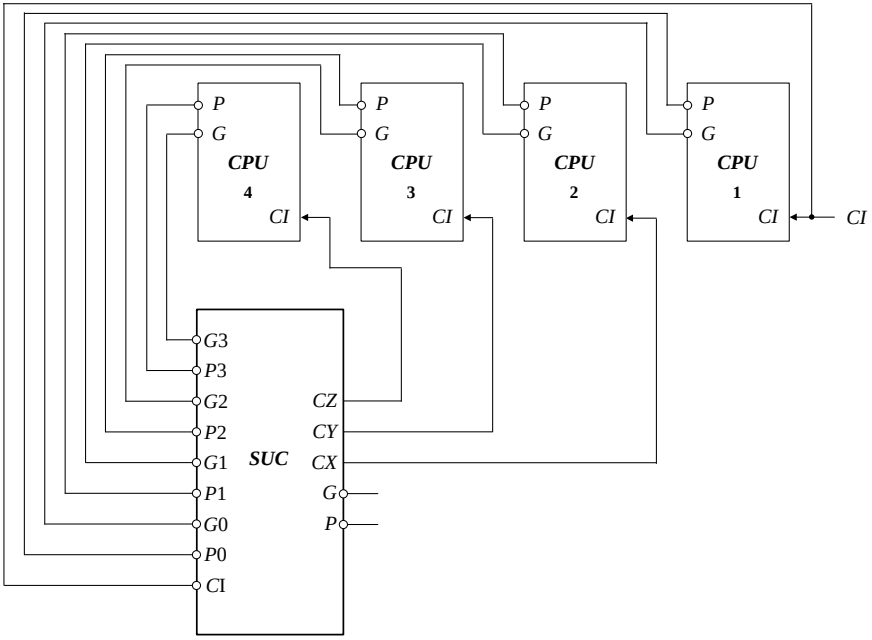


Рис. 2.23. Організація ланцюгів розповсюдження переносу шістнадцяти-розрядного ПЕ

Приклад 2.5. Розробити мікропрограму обчислення заданої функції для шістнадцятирозрядного блоку обробки даних.

$$F = 2X3 \& (X2 - X1)/2$$

Вихідні дані: $X1 = -3$, $X2 = 15$, $X3 = 7$.

Вважатимемо, що для подання операндів використовується доповнювальний код. Мікроалгоритм обчислення функції F показаний на рис. 2.24. В коментарях до кожної операторної вершини наведені значення управляючих сигналів ІС, *AA*, *BB*, *CI*, *SB*15, *SB*0, *SQ*15, *SQO*, що забезпечують виконання відповідної мікрооперації.

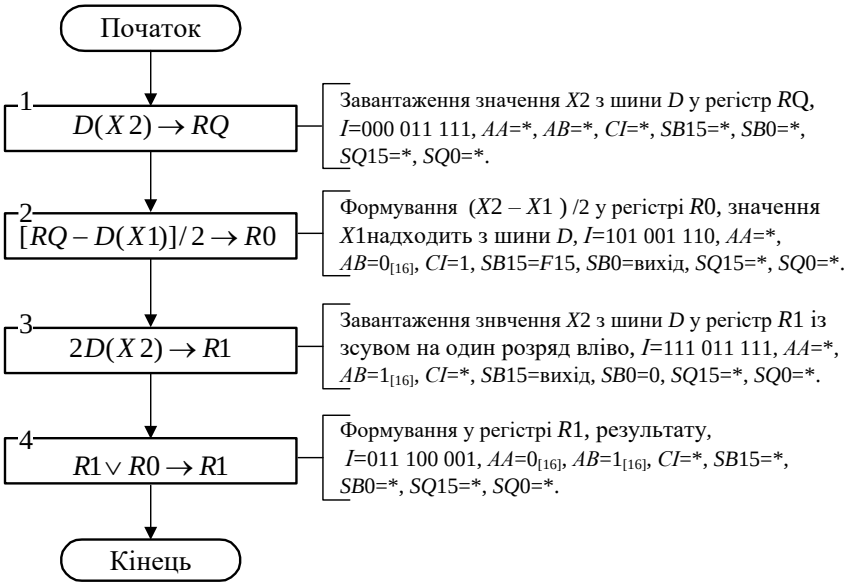


Рис. 2.24. Мікроалгоритм обчислення функції

Послідовність мікрокоманд, що реалізують заданий мікроалоритм та цифрова діаграма стану регістрів під час виконання мікропрограми відображені у табл. 2.25.

Розглянемо мікрооперацію $[RQ - D(X1)]/2 \to R0$, що виконується в другому такті. Джерелами даних, що надходять на входи *S* і *R* АЛБ (рис. 2.1), є регістр *RQ* і дані на вхідній шині *D*.

Відповідно до табл. 2.4. визначаємо розряди $I_2I_1I_0$ поля АЛБ_МІ мікрокоманди $I_2I_1I_0 = 110$. Мікрооперація віднімання, що здійснюється у цьому такті, визначається розрядами $I_5I_4I_3 = 001$ поля АЛБ_МІ мікрокоманди (табл. 2.3). при цьому встановлюється *CI*=1, що відповідає розрядам $I_{12}I_{11} = 01$ поля СУСЗ_МІ мікрокоманди (табл. 2.19). Результат, отриманий на виході *Y* АЛБ записується в регістр НОЗП за адресою, виставленою на шині *AB*. За умовою завдання результат має бути записаний у регістр *R*0, тому у полі *B* АЛБ_МІ мікрокоманди (табл. 2.3) розміщується адреса відповідного регістру у шістнадцятирічному поданні B = 0. При цьому кодуємо розряди $I_8I_7I_6$ поля АЛБ_МІ мікрокоманди (табл. 2.2), враховуючи те, що результат мікрооперації має бути зсунутий на один розряд вправо (що відповідає діленню на два за умовою завдання), отримуємо $I_8I_7I_6 = 101$.

При арифметичному зсуві вправо розряд, що звільнився, заповнюється знаком числа. Оскільки зсув відбувається на зсувачі ЗСВ, то з урахуванням можливого переповнювання розрядної сітки необхідно забезпечити $SB15 = F15 \oplus OVR$ (де, $F15$ – знак результату, *OVR* – ознака переповнювання).

Відповідну комутацію виводів зсувача та формування сигналу переносу *CO* забезпечує ІС ВР2. Подача нуля на вхід переносу ($CI = 0$) забезпечується встановленням розрядів поля СУСЗ_МС мікрокоманди $I_{12}I_{11} = 00$, подача одиниці ($CI = 1$) – встановленням $I_{12}I_{11} = 01$ поля СУСЗ_МІ мікрокоманди.

Якщо зсув не виконується, виводи ВР2, пов'язані із зсувачем, відключаються сигналом $\overline{SE} = 1$, що відповідає розрядам $I_{12}I_{11} = 01$ поля СУСЗ_МІ мікрокоманди (табл. 2.19). Під час виконання зсувів встановлюється $\overline{SE} = 0$.

Різні типи зсувів кодуються розрядами ($I_{10}..I_6$) поля СУСЗ_МІ мікрокоманди (табл. 2.19) відповідно до табл. 2.16. Арифметичним зсувам відповідають наступні управляючі сигнали, що надходять на входи ВР2: зсув вправо $I_{10}..I_6 = 01110$, вліво – $I_{10}..I_6 = 10010$.

Таблиця 2.25. Кодова карта

№ такту	Адреса [16]	Константа БОД		БОД													
				ВС1 (АЛБ)				ВР2 (СУСЗ)									
		D [16]	$\overline{OED}$	АЛБ_МІ [2]	*A* [16]	*B* [16]	$\overline{OEY}$	СУСЗ_МІ [2]	$\overline{EC}$	$\overline{EZ}$	$\overline{EN}$	$\overline{EV}$	$\overline{CEN}$	$\overline{CEM}$	$\overline{OECT}$	$\overline{SE}$
1	0000	000*F*	0	000 011 111	*	*	1	00 00000 000000	0	0	0	0	1	1	1	1
2	0001	*FFFD*	0	101 001 110	*	0	1	01 01110 000000	0	0	0	0	1	1	1	0
3	0002	0007	0	111 011 111	*	1	1	00 10010 000000	0	0	0	0	1	1	1	0
4	0003	****	1	011 100 001	0	1	1	00 00000 000000	1	1	1	1	1	1	1	1

Цифрова діаграма стану регістрів

№ такту	Вихідний стан				Інформація у приймачі результату, [16]	Мікрооперація
	Шина ***D***, [16]	***R0***, [16]	***R1***, [16]	***RQ***, [16]		
1	000*F*	****	****	****	$RQ := 000F$	$RQ := D(000F) \vee 0$
2	*FFFD*	****	****	000*F*	$R0 := 0009$	$R0 := SRA[RQ - D(FFFD) - 1 + 1(CI)]$
3	0007	0009	****	000*F*	$R1 := 000E$	$R1 := SLA[D(0007) \vee 0]$
4	****	0009	000*E*	000*F*	$R1 := 0008$	$R1 := R1 \vee R0$
			0008			Результат

Примітка: *SRA*, *SLA* – мнемонічне позначення зсувів арифметичних вправо та вліво відповідно.

2.5. Блоки мікропрограмного управління

Мікросхема К1804ВУ4 призначена для управління послідовністю вибірки мікрокоманд з пам'яті мікрокоманд (ПМК).Структура ІС К1804ВУ4 зображена на рис. 2.25.

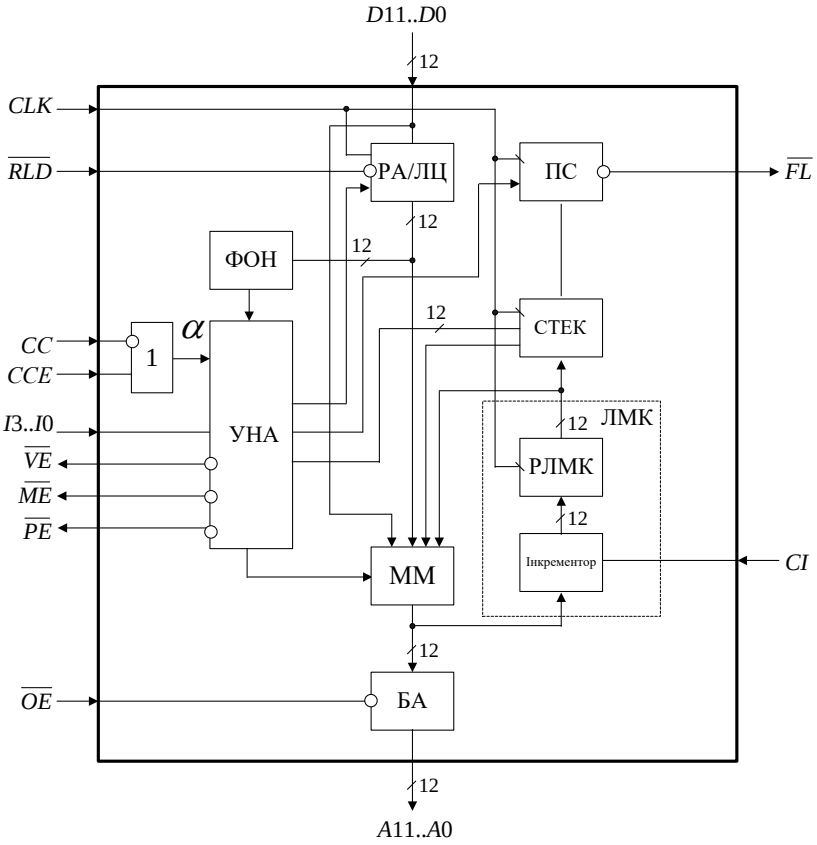


Рис 2.25. Структурна схема ІС К1804ВУ4

Мікросхема К1804ВУ4 складається з наступних функціональних частин:

УНА – схема управління вибіркою адреси наступної мікрокоманди;

РА/ЛЦ – регістр адреси/лічильник циклів;

ЛМК	–	лічильник мікрокоманд;
ПС	–	покажчик стека;
РЛМК	–	регістр лічильника мікрокоманд;
ММ	–	мультиплексор мікрокоманди;
ФОН	–	формувач ознаки нуля;
БА	–	буфер адреси.

Умовне графічне позначення ІС К1804ВУ4 показано на рис. 2.26, а призначення виводів наведене в табл. 2.26.

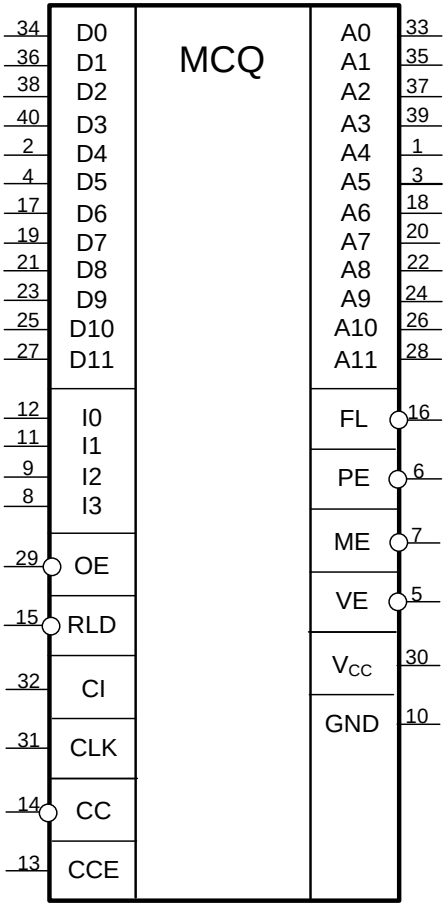


Рис. 2.26. Умовне графічне позначення ІС К1804ВУ4

ІС К1804ВУ4 виконує функції формувача адреси наступної мікрокоманди (ФАМ) і забезпечує формування дванадцятирозрядних адрес мікрокоманд у ПМК. Внаслідок чого ємність ПМК складає 4К 83-розрядних слів МК.

Виходи *A* адреси мікрокоманди (рис. 2.26) мають три стани (0, 1 і високоомний стан), що дозволяє, за необхідності, забезпечити зовнішній доступ до входів адреси ПМК, наприклад, для тестової перевірки послідовності мікрокоманд, виходу на початкову мікропрограму ініціалізації системи і таке інше.

Таблиця 2.26. Функціональне призначення виводів ІС К1804ВУ4

Номер виводу	Позначення виводів	Функціональне призначення
33, 35, 37, 39, 1, 3, 18, 20, 22, 24, 26, 28	*A*11 – *A*0	Виходи адреси мікрокоманд
34, 36, 38, 40, 2, 4, 17, 19, 21, 23, 25, 27	*D*11 – *D*0	Входи адреси розгалуження
5, 6, 7	*VE*, *PE*, *ME*	Виходи дозволу підключення до входів *D* Буферу *V*, Буферу *P* та Буферу *M* відповідно
12, 11, 9, 8	*I*3 – *I*0	Входи коду інформаційного слова мікроінструкції переходу
10	*GND*	Вивід живлення (загальний)
13	*CC*	Вхід умови
14	*CCE*	Вхід дозволу умови
15	*RLD*	Вхід дозволу запису в РА/ЛС
16	*FL*	Вихід ознаки заповнення стеку
29	*OE*	Вхід дозволу видачі адреси
30	*VCC*	Вихід живлення (+5*B*)
31	*CLK*	Вхід синхросигналу
32	*CI*	Вхід переносу в лічильник мікрокоманд

Принцип формування адреси наступної мікрокоманди

Спосіб формування адреси мікрокоманди визначається зовнішніми управляючими сигналами на входах *I*3 – *I*0, $\overline{CC}$, *CCE*

(рис. 2.25) і внутрішнім сигналом, який формується у схемі ФОН. Схема УНА, що є перетворювачем кодів, виробляє набір внутрішніх управляючих сигналів і зовнішні сигнали $\overline{PE}$, $\overline{VE}$ і $\overline{ME}$. Внутрішні управляючи сигнали виконують настройку блоків ІС схеми на виконання необхідний функцій. Зовнішні управляючі сигнали використовують для підключення до входів $D11 – D0$ ІС відповідних джерел інформації.

Мультиплексор ММ управляється внутрішніми сигналами, що надходять зі схеми УНА, та виконує функцію вибору одного з чотирьох джерел надходження адреси наступної мікрокоманди.

Таким чином, джерелами адреси наступної мікрокоманди є:

- входи $D11 – D0$,
- регістр адреси/лічильник циклів – РА/ЛЦ,
- СТЕК,
- лічильник мікрокоманд – ЛМК.

Обрана адреса через БА видається на виходи *A* за встановлення управляючого сигналу $\overline{OE} = 0$.

Дані в дванадцятирозрядний регістр РА/ЛЦ надходять з входів $D11 – D0$ та записуються за додатним перепадом синхросигналу *CLK*, якщо виконується певна управляюча мікроінструкція, що надходить на входи $I3 – I0$ ІС. Під час запису даних у РА/ЛЦ встановлюється управляючий сигнал $\overline{RLD} = 0$, при цьому дані у РА/ЛЦ можуть бути записані і незалежно від виконуваної мікроінструкції.

При виконанні відповідних мікроінструкцій ($I3 – I0$) вміст РА/ЛЦ зменшується на одиницю за додатним сигналом *CLK*, що дозволяє використовувати РА/ЛЦ в якості лічильника циклів. Ознака того, що вміст лічильника циклів дорівнює нулю – РА/ЛЦ = 0, формується у схемі ФОН.

Таким чином, у РА/ЛЦ надходить інформація з зовнішніх входів *D* (при умові встановлення $\overline{RLD} = 0$), і у подальшому використовується або в якості адреси наступної мікрокоманди, або як кількість повторів циклів.

Лічильник мікрокоманд складається з дванадцятирозрядного РЛМК та ІНКРЕМЕНТОРА, що є комбінаційною схемою додавання вхідного переносу *CI* до вхідного слова. Будь-яка адреса, з виходу мультиплексора ММ записується за додатним перепадом *CLK* в

регістр РЛМК, під час встановлення сигналу $CI = 0$, або збільшеною на одиницю, під час встановлення сигналу $CI = 1$. Таким чином наприкінці кожного такту у ЛМК формується адреса наступної мікрокоманди у вигляді $A_{i+1} = A_i + 1$, яка у наступному такті при виконанні лінійних переходів є джерелом адреси наступної мікрокоманди.

Стек складається з покажчика стеку (ПС) і накопичувача (СТЕК), що має п'ять регістрів (глибина стеку дорівнює п'яти). Схема ПС є реверсивним лічильником, який управляється внутрішніми сигналами і змінює свій стан за додатним перепадом *CLK*.

Стек організований за принципом *LI/FO* (*Last Input / First Output* – останній прийшов / перший обслугований). Схема ПС вказує на регістр, у який за останнім зверненням здійснювався запис даних.

Залежно від виконуваних мікроінструкцій (*I*3 –*I*0) стек працює в наступних режимах:

– *очищення стека* – встановлення покажчика стеку в нуль (ПС = 0);

– *збереження інформації* – ПС не змінює свого стану, з відповідного регістру накопичувача СТЕК (визначеному ПС) здійснюється зчитування даних;

– *завантаження стеку* – за додатним перепадом *CLK* значення ПС збільшується на одиницю в послідовності 0, 1, 2, 3, 4, 5, після чого здійснюється запис інформації в регістр накопичувача СТЕК; при заповненні стека (ПС = 5) встановлюється сигнал $\overline{FL} = 0$ і при подальшому завантаженню стеку ПС не змінює свого стану;

– *виштовхування із стеку* – відбувається зчитування інформації із стеку і зменшення ПС на одиницю в послідовності 5, 4, 3, 2, 1, 0; при встановленні ПС = 0 операція виштовхування із стеку приводить до зчитування невизначеної інформації, при цьому ПС не змінює свого стану.

ІС К1804ВУ4 дозволяє реалізувати 16 мікроінструкцій, які застосовуються для розгалуження мікропрограм і забезпечують:

– лінійні переходи – формування наступної адреси (інкремент);

– багаторазове повторення однієї й той самої адреси;

– умовні та безумовні переходи;

– організацію циклів;
– умовні та безумовні виклики мікропідпрограм.

Мікроінструкції є безумовними і умовними, тобто виконуваними при виконанні певної умови. В залежності від сигналів на входах $\overline{CC}$ і *CCE* формується ознака виконання умови $\alpha = \overline{CC} \vee CCE$ (рис. 2.25). Якщо умова виконується встановлюється $\alpha = 1$, інакше $\alpha = 0$. Таким чином умова виконується, якщо $\overline{CC} = 0$ або $CCE = 1$. Незалежно від значення сигналу на вході $\overline{CC}$, можна примусово забезпечити виконання умови $\alpha = 1$, встановив $CCE = 1$, тобто забезпечити безумовний перехід.

Характер переходів під час виконання мікроінструкцій наведено у табл. 2.27.

На базі ІС К1804ВУ4, призначеної для формування та управління послідовністю вибору мікрокоманд із ПМК можна побудувати БМУ, структурна схема якого зображена на рис. 2.27.

БМУ складається з наступних функціональних частин:

ФАМ – схема формування вибіркою адреси наступної мікрокоманди, реалізована на ІС ВУ4;
ПМК – пам'ять мікрокоманд;
РМК – регістр мікрокоманди;
МУ – мультиплексор вибору логічних умов;
ІНВЕРТОР – інвертор;
ШАР – дванадцяти розрядна шина адреси розгалуження;

три буфери джерел адрес мікрокоманд:

Буфер *M* – буфер регістра адреса;
Буфер *V* – буфер регістра вектора переривань;
Буфер *P* – буфер регістра константи.

Для управління БМУ у структурі мікрокоманди відведено двадцять три розряди (рис. 2.28):

P – дванадцятирозрядна частина поля констант МК (поле *P*), що містить адресу переходу або константу, наприклад, кількість повторень циклів;
ФАМ_ІС – чотирирозрядне інформаційне слово, що визначає мікроінструкцію переходу;
CCE – дозвіл аналізу умови;

COM	– інвертування сигналу логічної умови;
CI	– дозвіл формування адреси наступної МК;
$\overline{RLD}$	– дозвіл запису константи з поля Р МК в РА/ЛЦ ФАМ;
MS	– трирозрядне поле управління вибором входу МУ.

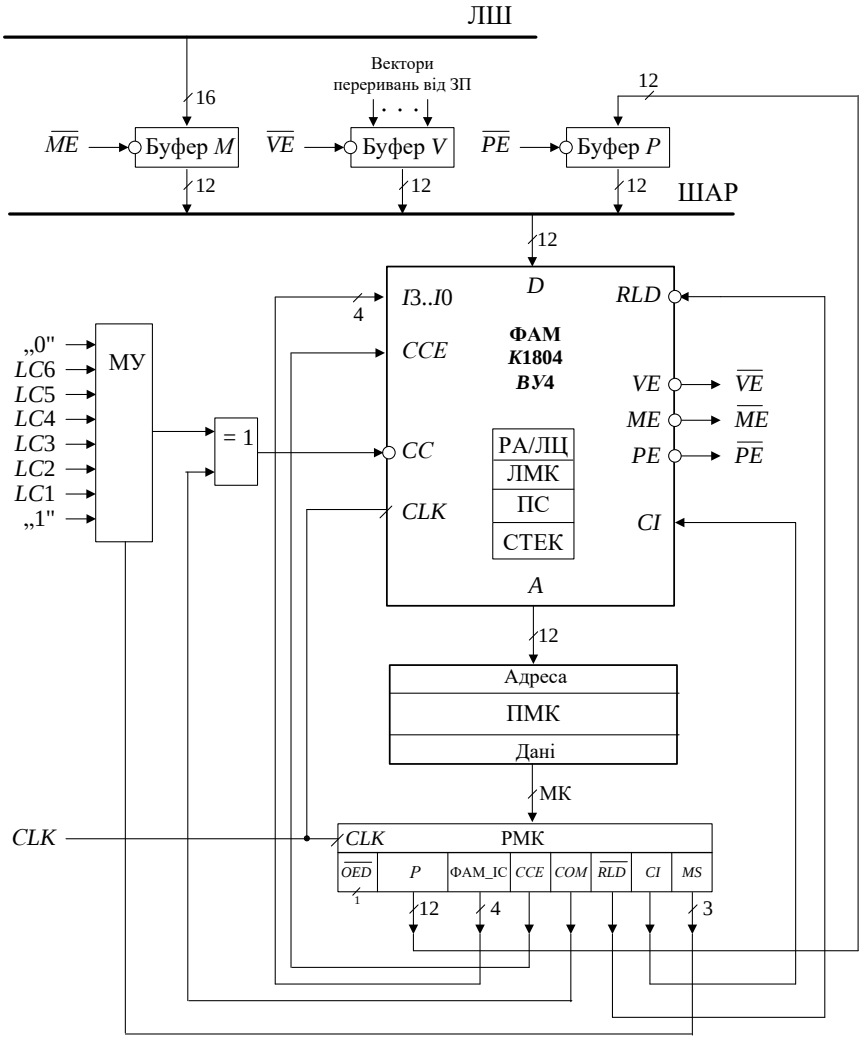


Рис 2.27. Структурна схема БМУ

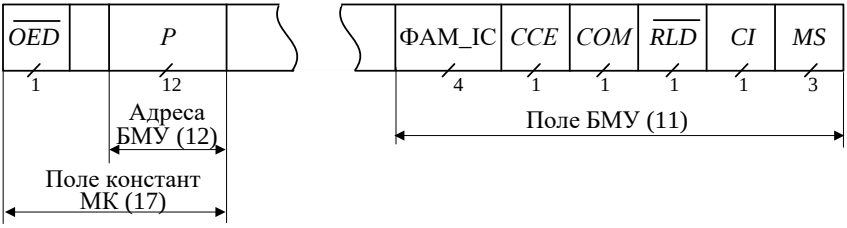


Рис. 2.28. Поля мікрокоманди, які використовуються для управління блоком мікропрограмного управління

Принцип функціонування БМУ

Буфери *M*, *V* і *P*, що мають виходи з трьома станами, призначені для видачі на ШАР дванадцятирозрядних адрес. Сигнали $\overline{ME}$, $\overline{VE}$ і $\overline{PE}$ що формує УНА (рис. 2.26) відкривають Буфери *M*, *V* і *P*, відповідно – $\overline{ME}=0$, $\overline{VE}=0$ і $\overline{PE}=0$. У поточному машинному такті може бути відкритий тільки один з буферів. Таким чином, якщо джерелом адреси наступної мікрокоманди у ІС ВУ4 є входи *D*11 – *D*0, то інформація на ці входи видається з одного з буферів *M*, *V* або *P* по ШАР.

Зазвичай Буфер *M* та сигнал $\overline{ME}=0$ використовується для прийому початкової адреси *M* мікропрограми, який визначається полем коду операції команди та надходить у БМУ з локальної шини МПС.

Через Буфер *V* ($\overline{VE}=0$) здійснюється прийом початкової адреси *V* мікропрограми обслуговування переривання від зовнішнього пристрою за певним вектором.

Сигнал $\overline{PE}$ управляє Буфером *P*, через який з поля *P* регістра мікрокоманди на шину ШАР, а далі на шину *D* схеми ФАМ поступає адреса розгалуження або параметр циклу.

Зовнішні логічні умови $LC1-LC6$ надходять на вхід МУ, що комутує ці умови, а також значення 0 та 1 на вихід. З виходу МУ логічний сигнал надходить на вхід елементу ВИКЛЮЧНЕ АБО, який виконує функцію керованого інвертора, і далі на вхід $\overline{CC}$ схеми ФАМ. Інвертуванням сигналу умови управляє сигнал *COM* поля БМУ мікрокоманди. При встановленні сигналу *COM* =1 логічна умова інвертується.

Слід зазначити, що зовнішні логічні умови, що надходять на входи $LC1 - LC6$ мультиплексора МУ, формуються на виході *CT* БОД (рис. 2.6).

Таким чином, БМУ дозволяє реалізувати такі типові управляючі конструкції, як безумовні і умовні переходи, цикли, мікропідпрограми.

Безумовний перехід означає передачу управління за певною адресою, не залежною від виконання логічних умов. Під час умовного переходу передача управління здійснюється за одною з двох або більше адрес залежно від виконання умов.

Для реалізації умовних переходів можна використовувати наступні мікроінструкції *CJP, CJV, JRP, CJPP* і *TWB*, а безумовних – *IZ, JMAP, LDCT* і *CONT* (табл. 2.27). Крім того, будь яку умовну мікроінструкцію можна перетворити на безумовну, якщо встановити на вході ВУ4 сигнал *CCE* = 1. При цьому, як вже було зазначено, інструкція виконуватиметься так, як встановиться сигнал $\alpha = 1$.

Циклічні конструкції мікропрограм зазвичай організовують з використанням мікроінструкцій *RFCT, RPCT, LOOP* і *TWB*. За цього інструкція *LOOP* забезпечує вихід з циклу за умовою, а інші – за нульовим вмістом лічильника циклів (РА/ЛЦ), тобто за кількістю повторень циклу. Для організації циклів можна також використовувати умовні переходи, вказавши в якості адреси переходу адресу мікрокоманди початку циклу.

Для переходу до мікропідпрограм зручно користуватися інструкціями *CJS, JSRP*, при виконанні яких адреса повернення із мікропідпрограми зберігається в стеку. Інструкція *CRTN* забезпечує повернення з мікропідпрограми за адресою, що вслід за цим виштовхується із стеку. Наявність стеку дозволяє організувати вкладені мікропідпрограми, тобто забезпечити звернення з однієї мікропідпрограми до іншої. Стек може берегти не більш ніж п'ять адрес повернення, що визначає максимальне число вкладень мікропідпрограм.

При застосуванні мнемонічного запису мікрокоманд, виконуваних схемою ФАМ можна використовувати мнемоніку найбільш розповсюджених логічних умов, що формуються СУСЗ:

`z` – заздалегідь хибна умова (умова ніколи не виконується);

`nz` – заздалегідь істинна умова (умова завжди виконується);

`zo`, `co`, `no`, `vo` – входи ознак стану *IZ, IC, IN, IV* СУСЗ;

`rm_z`, `rm_c`, `rm_n`, `rm_z` – розряди регістру стану *RM*: *MZ, MC, MN, MV*;

`rn_z`, `rn_c`, `rn_n`, `rn_z` – розряди регістру стану *RN*: *NZ, NC, NN, NV*;

`nxorv` – ознака $\overline{IN \oplus IV}$;

`zxorc` – ознака $\overline{IC \oplus IZ}$;

`l1`, `l2`, `l3`, `l4`, `l5`, `l6` – безпосередньо аналізувати інформацію на входах МУ: *L1, L2, L3, L4, L5, L6*;

`ct`, `rdm`, `rdd`, `int`, `irq0`, `irq1`, …, `irq7` – підключати до входів МУ сигнали, що генерують різні пристрої МПС: СУСЗ, пам'ять, зовнішні пристрої, сигнали вимоги переривань.

Для підключення інвертованих логічних умов можна застосовувати ту саму мнемоніку з префіксом `not`, наприклад, `not rm_c`.

Слід зазначити, якщо умова записується з префіксом `not`, то в поле *COM* мікрокоманди встановлюється значення 1.

Як уже було зазначено формуванням значення α на вході схеми УНА на основі логічної умови (ЛУ), що надійшла на вхід МУ, управляють поля мікрокоманди *CCE* та *COM* (рис. 2.25, рис. 2.27). Далі розглянуті декілька типових комбінацій цих полів, що забезпечують формування α.

– *CCE* = 0, *COM* = 1 $\Rightarrow$ перевірка умови, α = ЛУ, наприклад, `{cjp rm_c, addr;};`

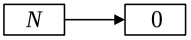
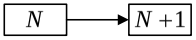
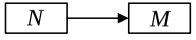
– *CCE* = 0, *COM* = 0 $\Rightarrow$ перевірка інвертованої умови, $\alpha = \overline{\text{ЛУ}}$, наприклад, `{cjp not rm_c, addr;};`

– *CCE* = 1, *COM* = * $\Rightarrow$ безумовний перехід, $\alpha = 1$, наприклад, `{cjp nz, addr;};`

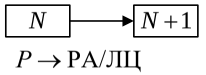
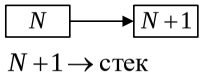
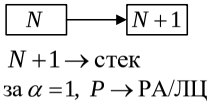
Реалізація мікрокоманд умовних та безумовних переходів наведена у прикладах 2.6 – 2.11. Для виводу коду умови через шину *CT* схеми СУСЗ в розглянутих прикладах встановлюється $\overline{OECT} = 0$, що у наведених фрагментах мікрокоманд умовно не показано.

В прикладах 2.12 – 2.16 представлено застосування управляючих конструкцій БМУ, організація циклів, реалізація мікропідпрограм.

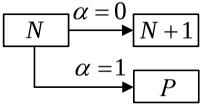
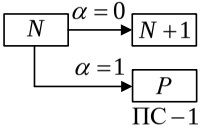
Таблиця 2.27. Управляючі конструкції БМУ

Мнемоніка мікрооперації	$(I_3..I_0)$	Сигнали та мікрооперації	Коментар	Графічна інтерпретація
1	2	3	4	5
Мікрокоманди безумовних переходів				
{jz;} – *перехід до нульової адреси*	0000	A:= 0000*h* $0 \rightarrow$ показчик стека ;	ЛМК встановлюється в значення ; відбувається; очистка стека	
{cont;} – *перехід до наступної адреси*	1110	A:= СМК;	Мікрокоманда призводить до інкременту ЛМК, і формування на виході *A* ФАМ адреси $A_{i+1} := A_i + 1$.	
{jmap;} – *безумовний перехід за адресою, що передається з Буферу M*	0010	$\overline{ME} = 0$; A:= *D*;	Використовується під час емуляції системи команд для переходу в ПМК до нової мікроподпрограми. При цьому на ЛШ має бути виставлена початкова адреса мікропідпрограми. Під час виконання МК формується сигнал $\overline{ME} = 0$, який відкриває Буфер *M*, і адреса нової мікропідпрограми з ЛШ надходить на вхід *D* ФАМ і далі на вихідну шину *A* (*A*:= *D*).	

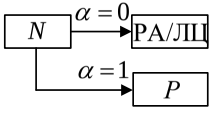
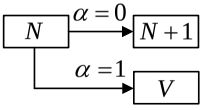
Продовження табл. 2.27.

1	2	3	4	5
{ldct val;} – *завантаження РА/ЛЦ та перехід до наступної адреси*	1100	$\overline{PE}=0$; $\overline{RLD}=0$; A:= СМК; $P \to$ РА/ЛЦ	Значення *val* у цьому випадку є вмістом поля *P* мікрокоманди. Під час виконання МК виробляється сигнал $\overline{PE}=0$, який відкриває Буфер *P*, і значення з поля *P* (кількість повторень циклу) надходить на вхід *D* ФАМ та записується в РА/ЛЦ. При цьому встановлюється $\overline{RLD}=0$.	
{push;} – *запис в стек*	0100	A:= ЛМК; ЛМК $\to$ стек ;	Мікрокоманда записує в стек наступну адресу, наприклад, якщо мікрокоманду розміщено в ПМК за адресою 2*d*3*h*, то під час її виконання в стек буде записано значення 2*d*4*h*.	
{push cond, val;} – *запис в стек та умовне завантаження РА/ЛЦ*	0100	$\overline{PE}=0$; A:= ЛМК; ЛМК $\to$ стек ; P $\to$ РА/ЛЦ,за $\alpha=0$ $\overline{RLD}=0$	Мікрокоманда записує в стек наступну адресу, завантажує в РА/ЛЦ значення *val*, якщо виконується умова (*cond* = 1). При цьому встановлюється $\overline{RLD}=0$. Якщо умова не виконується, то запис *val* в РА/ЛЦ не відбувається. Мікрокоманда рівноцінна двом мікрокомандам {push;} та {ldct val;}.	

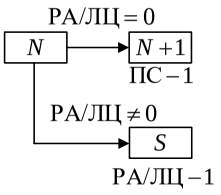
Продовження табл. 2.27.

1	2	3	4	5
Мікрокоманди умовних переходів				
{cjp cond, addr;} **{cjp cond, ml1;}** – *умовний перехід за адресою P*	0011	$\overline{PE} = 0$; за α = 0, A:= ЛМК; за α = 1, A:= D;	Мікрокоманда `cjp` (*Condition JumP*) дозволяє здійснювати розгалуження в мікропрограмах. Під час виконання мікрокоманди формується сигнал $\overline{PE} = 0$, який відкриває Буфер *P*, і значення *addr* з поля *P* мікрокоманди надходить на вхід *D* ФАМ. Якщо умова виконується (*cond*=1), то *A*:= *addr*, інакше *A*:= ЛМК.	
{cjpp cond, addr;} **{cjpp cond, ml1;}** – *умовний перехід за адресою P та виштовхування із стеку*	1011	$\overline{PE} = 0$; за α = 0, *A*:= ЛМК; за α = 1, *A*:= *D*, ПС=ПС – *1*;	Мікрокоманда `cjpp` (*Condition JumP and Pop*) відрізняється від МК `cjp` лише тим, що у разі виконання умови (*cond* = 1) додатково з верхівки стека виштовхується значення. Отже, цю мікрокоманду доцільно використовувати для виходу з циклу за умовою.	

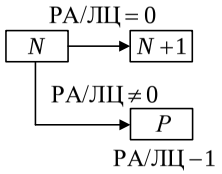
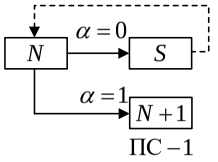
Продовження табл. 2.27.

1	2	3	4	5
{jrp cond, addr;} *– умовний перехід за адресою P або за адресою з РА/ЛЦ*	1011	$\overline{PE} = 0$; за α = 0, *A*:= РА/ЛЦ; за α = 1, *A*:= *D*;	Сигнал $\overline{PE} = 0$ відкриває Буфер *P*, значення *addr* з поля *P* мікрокоманди надходить на вхід *D* ФАМ. Якщо умова виконується (*cond* = 1), то A:=*addr*, і відбувається перехід за адресою, вказаною в мікрокоманді; інакше *A*:=(РА/ЛЦ), і відбувається перехід за адресою, яка зберігається в РА/ЛЦ.	
{cjv cond;} *– умовний перехід на підпрограму обробки переривання*	0110	$\overline{VE} = 0$; за α = 0, *A*:= ЛМК; за α = 1, *A*:= *D*;	Мікрокоманда `cjv` (*Condition Jump by Vector*) дозволяє здійснити перехід на мікропрограму обслуговування переривання (драйвер). На Буфер *V* надходять сигнали від зовнішніх пристроїв системи. Під час виконання цієї мікрокоманди формується сигнал $\overline{VE} = 0$, який відкриває Буфер *V*, і початкова адреса драйвера надходить на вхід *D* ФАМ. Якщо умова виконується (*cond* = 1), то A:= *V*, інакше – A:= ЛМК.	

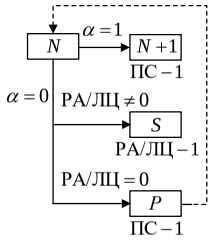
Продовження табл. 2.27.

1	2	3	4	5
Мікрокоманди для організації циклів				
{rfct;} – *повторити цикл (перехід за адресою з верхівки стека), якщо РА/ЛЦ ≠0*	1000	$\overline{PE} = 0$; за РА/ЛЦ ≠ 0, *A:*= Стек, РА/ЛЦ:=РА/ЛЦ – 1; за РА/ЛЦ = 0, *A*:=ЛМК, ПС:=ПС–1;	Використовують для багаторазового повторення певної частини мікропрограми, якщо відома кількість повторень. Кількість повторень перед початком виконання циклу записують в РА/ЛЦ за допомогою мікрокоманди `{push cond, val;}`. При цьому в стек заноситься адреса початку циклу. Після виконання тіла циклу, який слід завершити мікрокомандою `{rfct;}`, аналізується вміст РА/ЛЦ. Якщо РА/ЛЦ ≠ 0, здійснюється перехід на початок циклу (за адресою зі стека), а значення в РА/ЛЦ зменшується на 1. Повторення циклу відбувається, поки виконується РА/ЛЦ ≠ 0. Інакше (РА/ЛЦ = 0) здійснюється перехід до наступної мікрокоманди й виштовхування зі стека. Механізм виконання мікрокоманди `{rfct;}`такий, що декримент РА/ЛЦ відбувається після аналізу вмісту РА/ЛЦ. Тому завжди тіло циклу буде виконане на один раз більш ніж указано у параметрі `val`, вслід чого необхідна кількість циклів має бути зменшена на одиницю.	

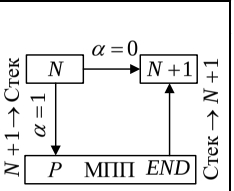
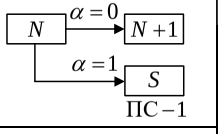
Продовження табл. 2.27.

1	2	3	4	5
{rpct addr;} – *повторити цикл (перехід за адресою P), якщо РА/ЛЦ = 0*	1001	$\overline{PE} = 0$; за РА/ЛЦ ≠ 0, *A*:= *D*, РА/ЛЦ:=РА/ЛЦ – 1; за РА/ЛЦ = 0, *A*:= ЛМК;	У мікрокоманді перехід відбувається за адресою *addr* з поля *P* мікрокоманди, що надійшла на вхід *D* ФАМ через Буфер *P* ($\overline{PE} = 0$). У іншому механізм роботи МК співпадає із МК `{rfct;}`, що розглянута раніш.	
{loop cond;} – *вихід з циклу за умовою*	1101	$\overline{PE} = 0$; за α = 0, *A*:= *S*; за α = 1, *A*:= ЛМК, ПС:=ПС – 1;	Мікрокоманда `{loop cond;}` використовується для завершення тіла циклу, якщо цикл має заздалегідь невідому кількість повторень. Якщо умова *cond* = 0 , то здійснюється перехід на початок циклу (за адресою збереженою у верхівці стеку), інакше здійснюється перехід до наступної мікрокоманди, що розташована наступною після МК завершення циклу і виштовхування зі стека адреси початку циклу.	

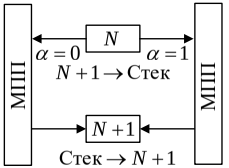
Продовження табл. 2.27.

1	2	3	4	5
{twb cond, addr;} *– розгалуження на три напрями*	1111	$\overline{PE} = 0$; за (α = 0) & (РА/ЛЦ ≠ 0), *A*:= Стек, РА/ЛЦ:=РА/ЛЦ – 1; за (α = 0) & (РА/ЛЦ = 0), *A*:= *D*, ПС:=ПС–*1;* за α = 1, *A*:= ЛМК, ПС:=ПС–*1;*	Під час виконання мікрокоманди `{twb cond, addr;}` аналізуються дві умови: *cond*, що явно вказана в мікрокоманді, та вміст РА/ЛЦ, що порівнюється з нулем. Якщо виконуються дві умови РА/ЛЦ≠0 і $cond = 0$, то здійснюється перехід за адресою з верхівки стека А:= Стек і вміст РА/ЛЦ зменшується на одиницю. Якщо РА/ЛЦ=0 і $cond = 0$, то здійснюється перехід за адресою *addr* з поля *P* мікрокоманди, що надійшла на вхід *D* ФАМ через Буфер *P* ($\overline{PE} = 0$), і виштовхування адреси повернення на початок циклу зі стека. Якщо виконується умова $cond = 1$, то здійснюється перехід до наступної після виконання циклу адреси А:=ЛМК.	

Продовження табл. 2.27.

1	2	3	4	5
Мікрокоманди для роботи з мікропідпрограмами				
{cjs cond, addr;} – *виклик мікропрограми за умовою*	0001	$\overline{PE}=0$; за α = 0, *A*:= ЛМК; за α = 1, *A*:= *D*, ЛМК → Стек ;	Мікрокоманда {cjs cond, addr;} (*Condition Jump to Subroutine*) забезпечує реалізацію механізму виклику мікропідпрограми за умовою, відбувається перехід на мікропідпрограму, що знаходиться в ПМК за адресою *addr*, якщо вказана умова виконується (*cond* =1). При цьому в стек записується адреса повернення (тобто адреса наступної за cjs мікрокоманди), а на вихідну шину *A* ФАМ надходить *addr* (з поля *P* через Буфер *P*), інакше здійснюється перехід до наступної мікрокоманди.	
{crtn cond;} – *умовне повернення з мікропідпрограми*	1010	$\overline{PE}=0$; за α = 0, *A*:= ЛМК; за α = 1, *A*:= Стек, ПС:=ПС–*1;*	Мікрокоманда {crtn cond;} (*Condition ReTurN*) забезпечує повернення з мікропідпрограми. Якщо виконується умова *cond* =1, то виконання мікропідпрограми припиняється, здійснюється повернення в основну мікропрограму за адресою з верхівки стеку (*A*:= Стек) і виштовхування зі стека. Інакше відбувається перехід до наступної мікрокоманди мікропідпрограми.	

Продовження табл. 2.27.

1	2	3	4	5
{jsrp cond,addr;} – *перехід до однієї з двох мікропідпрограм*	0101	$\overline{PE} = 0$; за α = 0, *A:*= РА/ЛЦ, ЛМК → Стек ; за α = 1, *A*:= *D;*	Під час виконання мікрокоманди {jsrp cond,addr;} в стек записується адреса повернення з МПП (тобто адреса наступної за JSRP мікрокоманди) та здійснюється виклик однієї з двох мікропідпрограм – мікропідпрограми, що розміщена за адресою *addr*, якщо виконується умова $cond = 1$. (за цього значення *addr* з поля *P* МК через Буфер *P* надходить на вихід *A* ФАМ, під час встановлення сигналу $\overline{PE} = 0$), або мікропідпрограми, що розміщена за адресою з РА/ЛЦ (за цього A:=РА/ЛЦ).	

Примітки: *N* – адреса виконуваної МК; *cond* – логічна умова; *addr* – адреса переходу; інші позначення відповідають рис. 2.27.

Приклад 2.6. Якщо результат обчислення суми у поточному такті від'ємний (*NO* = 1), перейти за адресою 0007*H* (за міткою `m1`), при цьому виконати виштовхування даних зі стеку, інакше продовжити обчислення.

Мікрокоманда

Адреса [16]	*D* [16]	БМУ_МІ [2]	*CCE*	*COM*	*CI*	$\overline{RLD}$	*MS*	СУСЗ_МІ $I_5 - I_0$	*P*	*CT*
0000	0007	1011	0	1	1	1	1	111110	0007	*IN*

Мнемонічний запис

```
          link l1:ct;
000       {add r1,r1,r0;cjpp no, m1;}
...       {}
007  m1   {}
```

Приклад 2.7. Виконати безумовний перехід на мітку `m1`, інакше перейти за адресою з РА/ЛЦ.

Мікрокоманда

Адреса [16]	*D* [16]	БМУ_МІ [2]	*CCE*	*COM*	*CI*	$\overline{RLD}$	*MS*	СУСЗ_МІ $I_5 - I_0$	РА/ЛЦ	*P*	*CT*
0000	0007	0111	1	*	1	1	*	000000	000*A*	0007	*

Мнемонічний запис

```
0000      {jrp nz,m1;}
          {}
0007 m1   {}
          ...
000A      {}
```

Приклад 2.8. Якщо результат виконання підсумовування не дорівнює нулю то перейти на мікропідпрограму за адресою, що надійшла з Буферу *V*.

Мікрокоманда

Адреса [16]	*D* [16]	БМУ_МІ [2]	*CCE*	*COM*	*CI*	$\overline{RLD}$	*MS*	СУСЗ_МІ $I_5 - I_0$	*V*	*CT*
0000	0007	0110	0	0	1	1	2	110100	0007	*IZ*

Мнемонічний запис

```
      link l2:ct;
0000  {add r1,r1,z;cjv not zo;}
```

Приклад 2.9. Завантажити у РА/ЛЦ значення, якщо виконується умова CT = 1.

Мікрокоманда

Адреса [16]	*D* [16]	БМУ_МІ [2]	*CCE*	*COM*	*CI*	$\overline{RLD}$	*MS*	СУСЗ_МІ $I_5 - I_0$	*P*	*CT*
0000	0006	0100	1	0	1	0	1	000100	0006	*NZ*

Мнемонічний запис

```
      link l1:ct;
0000  {push ct,6;}          \ якщо CT=1, то РА/ЛЦ:=6
```

Приклад 2.10. Якщо отримано нульовий результат (ZO = 1), то перейти на мітку `ml1`, інакше продовжити обчислення.

Мікрокоманда

Адреса [16]	*D* [16]	БМУ_МІ [2]	*CCE*	*COM*	*CI*	$\overline{RLD}$	*MS*	СУСЗ_МІ $I_5 - I_0$	*P*	*CT*
0000	0007	0011	0	1	1	1	1	110100	0007	*IZ*

Мнемонічний запис

```
          link l1:ct;
0000      {cjp zo,ml1;}
          {}
0007 ml1  {}
```

Приклад 2.11. Якщо *NV* = 1, то здійснюється виклик мікропідпрограми за адресою *SUB*.

Мікрокоманда

Адреса [16]	*D* [16]	БМУ_МІ [2]	*CCE*	*COM*	*CI*	$\overline{RLD}$	*MS*	СУСЗ_МІ $I_5 - I_0$	*P*	*CT*
0000	0007	0001	0	1	1	1	4	000110	0007	*NV*

Мнемонічний запис

```
         link l4:ct;
0000     {cjs rn_v, sub;}     \ перевірка логічної умови
                              \ якщо NV = 1 – перехід на МПП
          ...
0007 sub {}                   \ початок МПП SUB
```

Приклад 2.12. Організувати цикл із заданою кількістю повторень можна в такий спосіб:

Мікрокоманда

Адреса [16]	*D* [16]	БМУ_МІ [2]	*CCE*	*COM*	*CI*	$\overline{RLD}$	$\overline{OECT}$	*MS*	СУСЗ_МІ $I_5 - I_0$	Коментарі		
										P	Стек	*CT*
0000	000*C*	0100	1	*	1	0	1	*	000000	000*C*	0001	*
0001	****	1110	*	*	1	1	1	*	000000	****	0001	*
...												*
0009	000*F*	1011	0	1	1	1	0	1	101010	000*F*	0000	*MC*
...												
000E	****	1000	*	*	1	1	1	*	000000	****	0001	*
000F	****	1110	*	*	1	1	1	*	000000	****	****	*

Мнемонічний запис

```
          link l1:ct;
0000      {push nz,12;}        \ кількість повторень циклу
0001      {}                   \ початок циклу
            ...
0009      {cjpp rm_c,mex}      \ перевірка логічної умови, якщо МС = 1 – вихід з циклу
            ...
000E      {rfct;}              \ кінець циклу
000F  mex {}                   \ продовження мікропрограми
```

Приклад 2.13. Організувати цикл із виходом за умовою можна у такий спосіб:

Мікрокоманда

Адреса [16]	*D* [16]	БМУ_МІ [2]	*CCE*	*COM*	*CI*	$\overline{RLD}$	$\overline{OECT}$	*MS*	СУСЗ_МІ $I_5 - I_0$	Стек	*CT*
0000	****	0100	0	*	1	1	1	*	000000	0001	*
0001	****	1110	*	*	1	1	1	*	000000	0001	*
. . .											
000E	****	1101	0	0	1	1	0	3	101111	0001	*MN*
000F	****	1110	*	*	1	1	1	*	000000	****	*

Мнемонічний запис

```
        link l3:ct;
0000    {push;}                  \ завантаження в стек адреси початку циклу
0001    {}                       \ початок циклу
         ...
000E    {loop not rm_n;}         \ перевірка логічної умови: якщо MN = 0 – вихід з циклу,
                                 \ інакше – повернення на адресу із стеку
000F    {}                       \ продовження мікропрограми
```

Приклад 2.14. Розробити мікропрограму для обчислення заданої функції:

$$D = C/2 + 2A(B+1)$$

Виконання завдання:

Алгоритм обчислення функції наведений на рис. 2.29. Цифрова діаграма стану регістрів наведена у табл. 2.28. Мікропрограма налагоджена у моделюючому комплексі COMPLEX зображена на рис. 2.30.

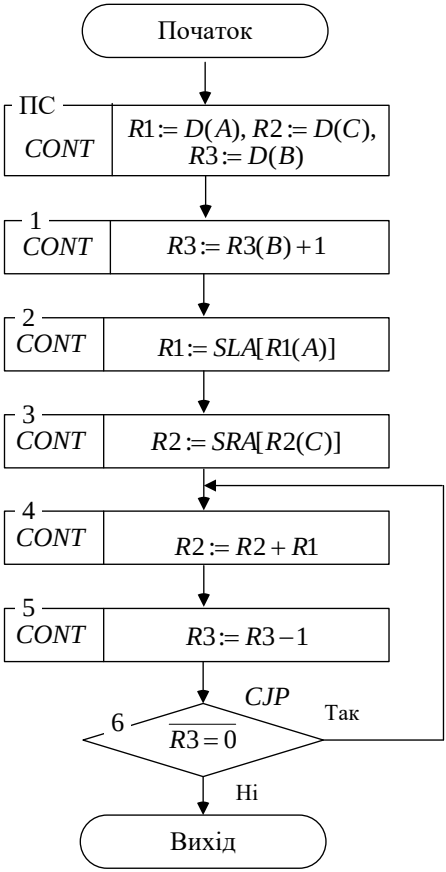
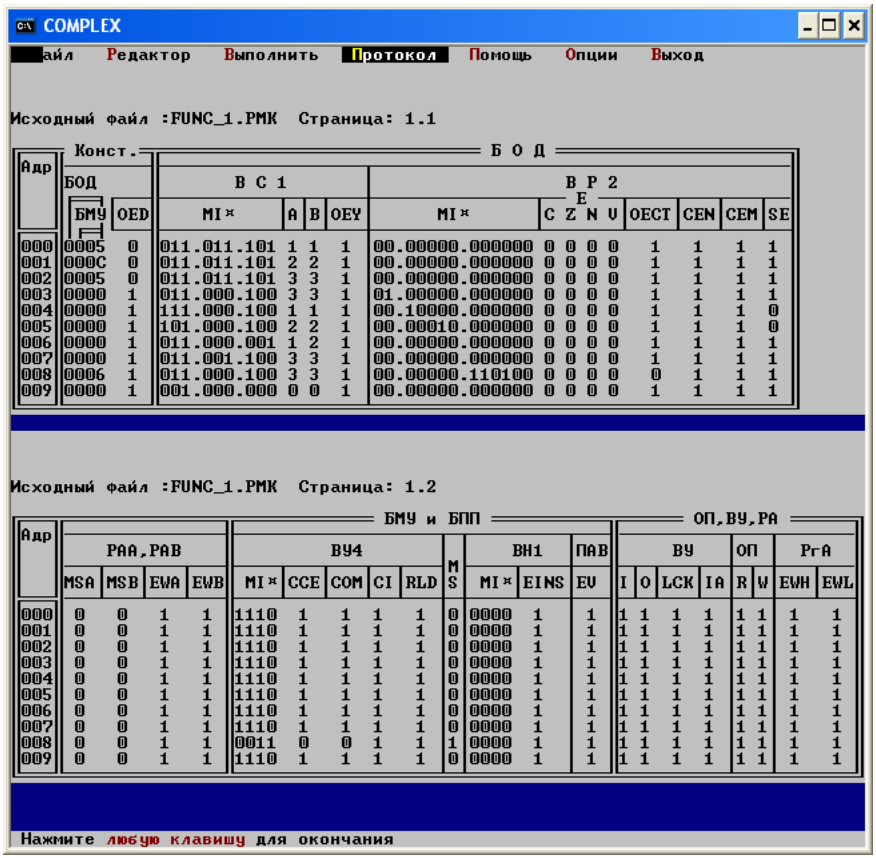


Рис. 2.29. Алгоритм обчислення функції

Таблиця 2.28. Цифрова діаграма стану регістрів

№ такту	***R*1 (*A*)**	***R*2(*C*)**	***R*3(*B*)**	***ZO***	**Коментарі**
ПС	0005	000C	0005	0	Завантаження даних
1			0006		$R3 := R3 + 1$ $(B + 1)$
2	000A				$R2 := l[R2].0$ $(2A)$
3		0006			$R1 := 0.r[R1]$ $(C/2)$
4(1)		0010			$R2 := R2 + R1$ $(A+C)$
5(1)			0005		$R3 := R3 - 1$ $(B - 1)$
6(1)				0	$ZO = 0$ $(B \neq 0)$
4(2)		001A			$R2 := R2 + R1$ $(A+C)$
5(2)			0004		$R3 := R3 - 1$ $(B - 1)$
6(2)				0	$ZO = 0$ $(B \neq 0)$
4(3)		0024			$R2 := R2 + R1$ $(A+C)$
5(3)			0003		$R3 := R3 - 1$ $(B - 1)$
6(3)				0	$ZO = 0$ $(B \neq 0)$
4(4)		002E			$R2 := R2 + R1$ $(A+C)$
5(4)			0002		$R3 := R3 - 1$ $(B - 1)$
6(4)				0	$ZO = 0$ $(B \neq 0)$
4(5)		0038			$R2 := R2 + R1$ $(A+C)$
5(5)			0001		$R3 := R3 - 1$ $(B - 1)$
6(5)				0	$ZO = 0$ $(B \neq 0)$
4(6)		0042			$R2 := R2 + R1$ $(A+C)$
5(6)			0000		$R3 := R3 - 1$ $(B - 1)$
6(6)				1	$ZO = 1$ $(B = 0)$
7					Кінець обчислення



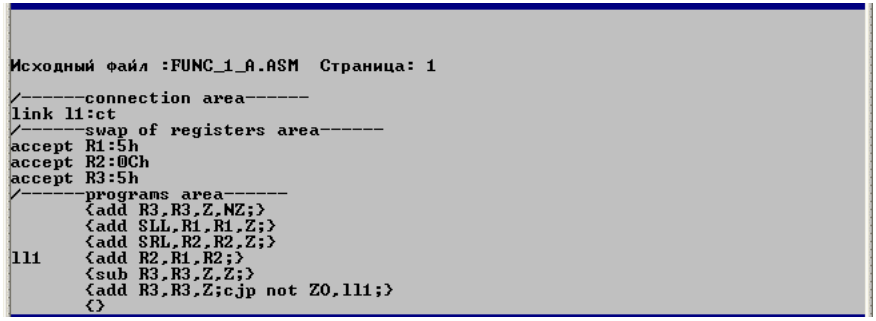


Рис. 2.30. Мікропрограма обчислення функції

Приклад 2.15. Розробити мікропрограму для блоку обробки даних, що реалізує заданий мікроалгоритм (рис. 2.31).

Виконання завдання:

На вихідному мікроалгоритмі (рис. 2.31) біля кожної вершини указані адреси розміщення відповідних мікрокоманд у ПМК, а також указані виконувані мікроінструкції для ІС К1804ВУ4, що забезпечують формування адрес мікрокоманд. Адреса першої вершини може бути довільною. Кодова карта з відповідними коментарями наведена у табл. 2.29. Кожній вершині мікроалгоритму відповідає рядок табл. 2.9, номер якої співпадає з номером вершини. Адреси та дані подані у шістнадцятирічній системі числення. Розряди мікрокоманди, значення яких можуть буди довільними, позначені символом «*».

У стовпці *LC* показані логічні умови, що впливають на формування адреси наступної мікрокоманди. Під час налагодження мікропрограми слід перевіряти правильність переходів, як за нульовим значенні умови, так і за одиничним.

На кодовій карті умовно не показані поля мікрокоманди, що не змінюються під час виконання мікропрограми.

Даний приклад ілюструє виконання кожної з шістнадцяти мікроінструкцій, що реалізує БМУ.

Умовні інструкції у вершинах 4, 5, 6, 8, 10, 15, 24 мікроалгоритму відповідають безумовному переходу. Безумовний перехід забезпечується поданням одиничного сигналу на вхід $CCE = 1$ (табл. 2.29). Слід зазначити, що безумовний перехід можна реалізувати також шляхом підключення до входу СС виходу *L*7 мультиплексора, що відповідає логічній одиниці. За цього у полях мікрокоманди треба встановити наступні сигнали: MS = 7[16] – номер виходу мультиплексора, $CCE = 0$, $COM = 1$(або $COM = 0$, якщо умова має бути інвертована) та $\overline{OECT} = 0$. Такий спосіб застосований під час кодування мікрокоманд на рис. 2.32.

Мікропрограма у машинних кодах та із застосуванням символічного мікроасемблеру, налагоджена у моделюючому комплексі *COMPLEX*, зображена на рис. 2.32, *а, б* відповідно.

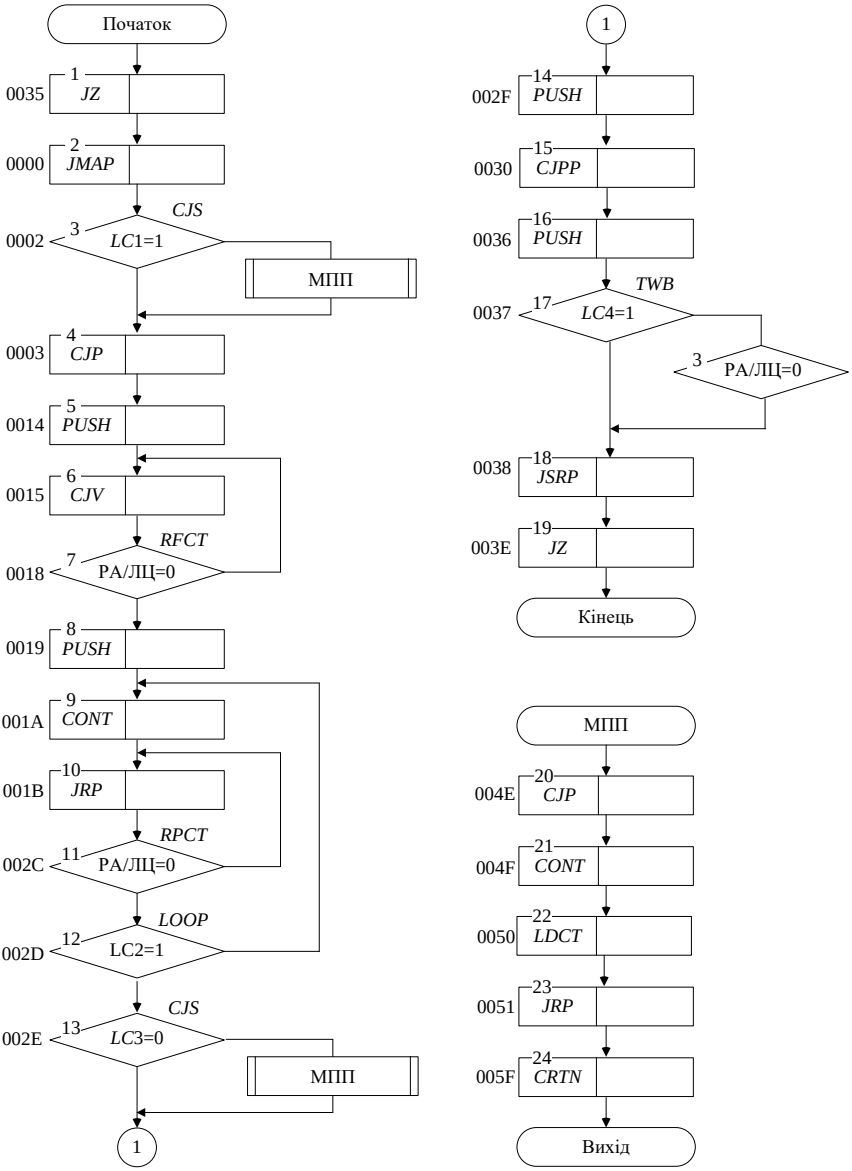


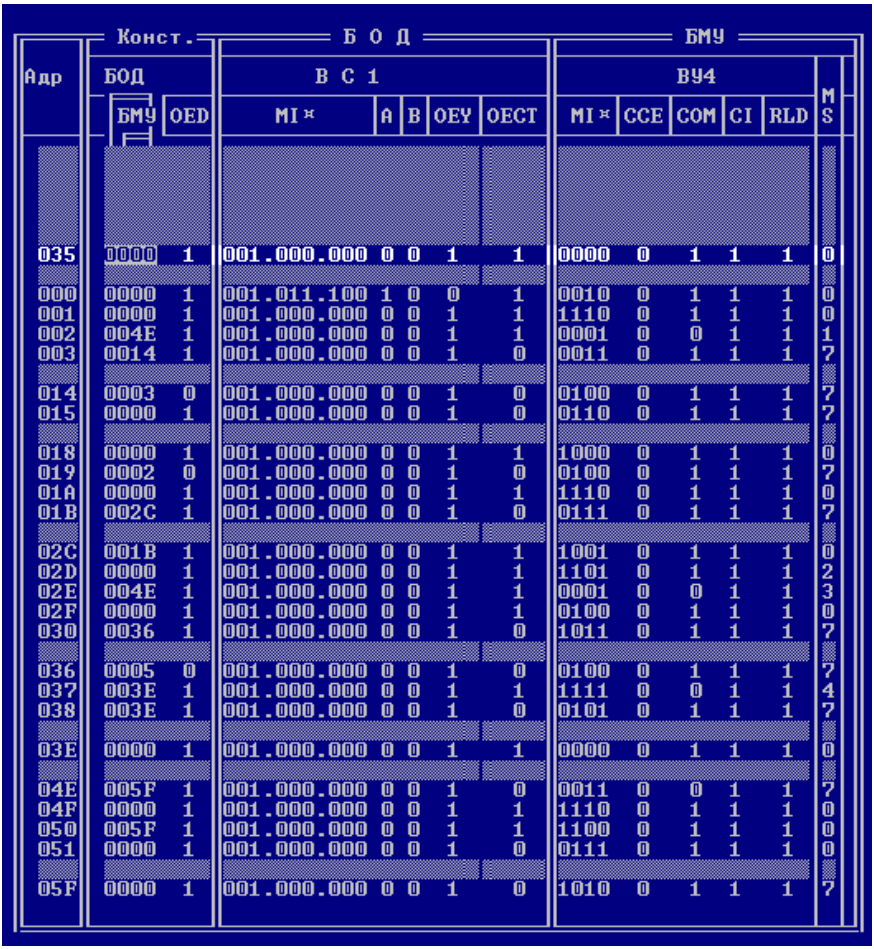
Рис. 2.31. Вихідний мікроалгоритм

Таблиця 2.29. Кодова карта

Мікрокоманда											**Коментарі**		
№ пп.	**Мнємоніка**	**Адреса** [16]	**Константа БОД**		**ВУ4 (БМУ)**								
			D [16]	$\overline{OED}$	БМУ_МІ [2]	*CCE*	*COM*	*CI*	$\overline{RLD}$	*MS*	*P* (*M*,*V*)	*ПС: СТЕК*	*LC*
1	*JS*	035	0000	1	0000	*	*	1	1	0	*	0:*	*
2	*JMAP*	000	0000	1	0010	*	*	1	1	*	0002(*M*)		*
3	*CJS*	002	004E	1	0001	0	0	1	1	1	004E	1:003	$\overline{L1=0}$
4	*CJP*	003	0014	1	0011	1	1	1	1	*	0014		1(*NZ*)
5	*PUSH*	014	0003	1	0100	1	1	1	0	*	0003	1:015	1(*NZ*)
6	*CJV*	015	0000	1	0110	1	1	1	1	*	0018(*V*)		1(*NZ*)
7	*RFCT*	018	0000	1	1000	*	1	1	1	*	*	0:*	РА/ЛЦ=0
8	*PUSH*	019	0002	1	0100	1	1	1	1	*	0002	1:01A	1(*nz*)
9	*CONT*	01A	0000	1	1110	*	1	1	1	*	*		*
10	*JRP*	01B	002C	1	0111	1	1	1	1	*	002C		1(*NZ*)
11	*RPCT*	02C	001B	1	1001	*	1	1	1	*	001B		РА/ЛЦ=0
12	*LOOP*	02D	0000	1	1101	0	1	1	1	2	*	0:*	*L*2=1

Продовження таблиці 2.29.

Мікрокоманда											Коментарі		
№ пп.	Мнємоніка	Адреса [16]	Константа БОД		ВУ4 (БМУ)								
			D [16]	$\overline{OED}$	БМУ_МІ [2]	*CCE*	*COM*	*CI*	$\overline{RLD}$	*MS*	*P* (*M*,*V*)	*ПС: СТЕК*	*LC*
13	*CJS*	02E	04E	1	0001	0	0	1	1	3	04E	1:02F	$\overline{L3=0}$
14	*PUSH*	02F	0000	1	0100	*	1	1	0	*	0000	1:030	*
15	*CJPP*	030	0036	1	1011	1	1	1	1	*	0036	0:*	1(*NZ*)
16	*PUSH*	036	0005	1	0100	0	1	1	0	7	0005	1:037	*L*7
17	*TWB*	037	003E	1	1111	0	0	1	1	4	003E	0:*	$\overline{L4=0}$
18	*JSRP*	038	003E	1	0101	0	1	1	1	7	003E	1:039	*L*7
19	*JZ*	038	0000	1	0000	*	*	1	1	0	*	0:*	*
20	*CJP*	04E	005F	1	0011	0	0	1	1	7	005F		$\overline{1(NZ)}$
21	*CONT*	04F	0000	1	1110	0	1	1	1	*	*		*
22	*LDCT*	050	005F	1	1100	1	1	1	0	*	005F		*
23	*JRP*	051	004F	1	0111	0	1	1	1	0	004F		0(*Z*)
24	*CRTN*	05F	0000	1	0111	1	1	1	1	*	*	0:*	1(*NZ*)



Конст.	БОД		ВС1					БМУ					
Адр	БОД БМУ	OED	MI*	A	B	OEY	OECT	ВУ4 MI*	CCE	COM	CI	RLD	MS
035	0000	1	001.000.000	0	0	1	1	0000	0	1	1	1	0
000	0000	1	001.011.100	1	0	0	1	0010	0	1	1	1	0
001	0000	1	001.000.000	0	0	1	1	1110	0	1	1	1	0
002	004E	1	001.000.000	0	0	1	1	0001	0	0	1	1	1
003	0014	1	001.000.000	0	0	1	0	0011	0	1	1	1	7
014	0003	0	001.000.000	0	0	1	0	0100	0	1	1	1	7
015	0000	1	001.000.000	0	0	1	0	0110	0	1	1	1	7
018	0000	1	001.000.000	0	0	1	1	1000	0	1	1	1	0
019	0002	0	001.000.000	0	0	1	0	0100	0	1	1	1	7
01A	0000	1	001.000.000	0	0	1	1	1110	0	1	1	1	0
01B	002C	1	001.000.000	0	0	1	0	0111	0	1	1	1	7
02C	001B	1	001.000.000	0	0	1	1	1001	0	1	1	1	0
02D	0000	1	001.000.000	0	0	1	1	1101	0	1	1	1	2
02E	004E	1	001.000.000	0	0	1	1	0001	0	0	1	1	3
02F	0000	1	001.000.000	0	0	1	1	0100	0	1	1	1	0
030	0036	1	001.000.000	0	0	1	0	1011	0	1	1	1	7
036	0005	0	001.000.000	0	0	1	0	0100	0	1	1	1	7
037	003E	1	001.000.000	0	0	1	1	1111	0	0	1	1	4
038	003E	1	001.000.000	0	0	1	0	0101	0	1	1	1	7
03E	0000	1	001.000.000	0	0	1	1	0000	0	1	1	1	0
04E	005F	1	001.000.000	0	0	1	0	0011	0	0	1	1	7
04F	0000	1	001.000.000	0	0	1	1	1110	0	1	1	1	0
050	005F	1	001.000.000	0	0	1	1	1100	0	1	1	1	0
051	0000	1	001.000.000	0	0	1	0	0111	0	1	1	1	0
05F	0000	1	001.000.000	0	0	1	0	1010	0	1	1	1	7

Рис. 2.32. Результати моделювання: а –мікропрограма в машинних кодах, б – мікропрограма у мнемокодах.

```
Исходный файл :PROG1.ASM  Страница: 1

        link m:12,11,10,9,8,7,6,5,4,3,2,1
        link v:z,z,z,z,z,z,z,z,nz,nz,z,z,z
        link l2:nz
        accept r1:0005
        org 0035h
        {jz;}
        org 0000h
        {or nil,r1,z;oey;jmap;}
        {cont;}
        {cjs not l1,mpp1;}
        {cjp nz, lll;}
        org 0014h
lll     {push nz,0003;}
        {cjv nz;}
        org 0018h
l12     {rfct;}
        {push nz,0002;}
        {cont;}
l14     {jrp nz,l13;}
        org 002ch
l13     {rpct l14;}
        {loop l2;}
        {cjs not l3,mpp1;}
        {push;}
        {cjpp nz,l15;}
        org 0036h
l15     {push nz,0005h;}
        {twb not l4,003eh;}
        {jsrp nz,l16;}
        org 003eh
l16     {jz;}
        org 004eh
mpp1    {cjp not nz,l17;}
        {cont;}
        {ldct 005fh;}
        {jrp z,0000;}
        org 005fh
l17     {crtn nz;}
```

Продовження рисунку 2.32

Приклад 2.16. Розробити мікропрограму для блоку обробки даних, що реалізує заданий мікроалгоритм (рис. 2.33).

Виконання завдання:

Мікропрограма у машинних кодах, налагоджена у моделюючому комплексі *COMPLEX*, зображена на рис. 2.34.

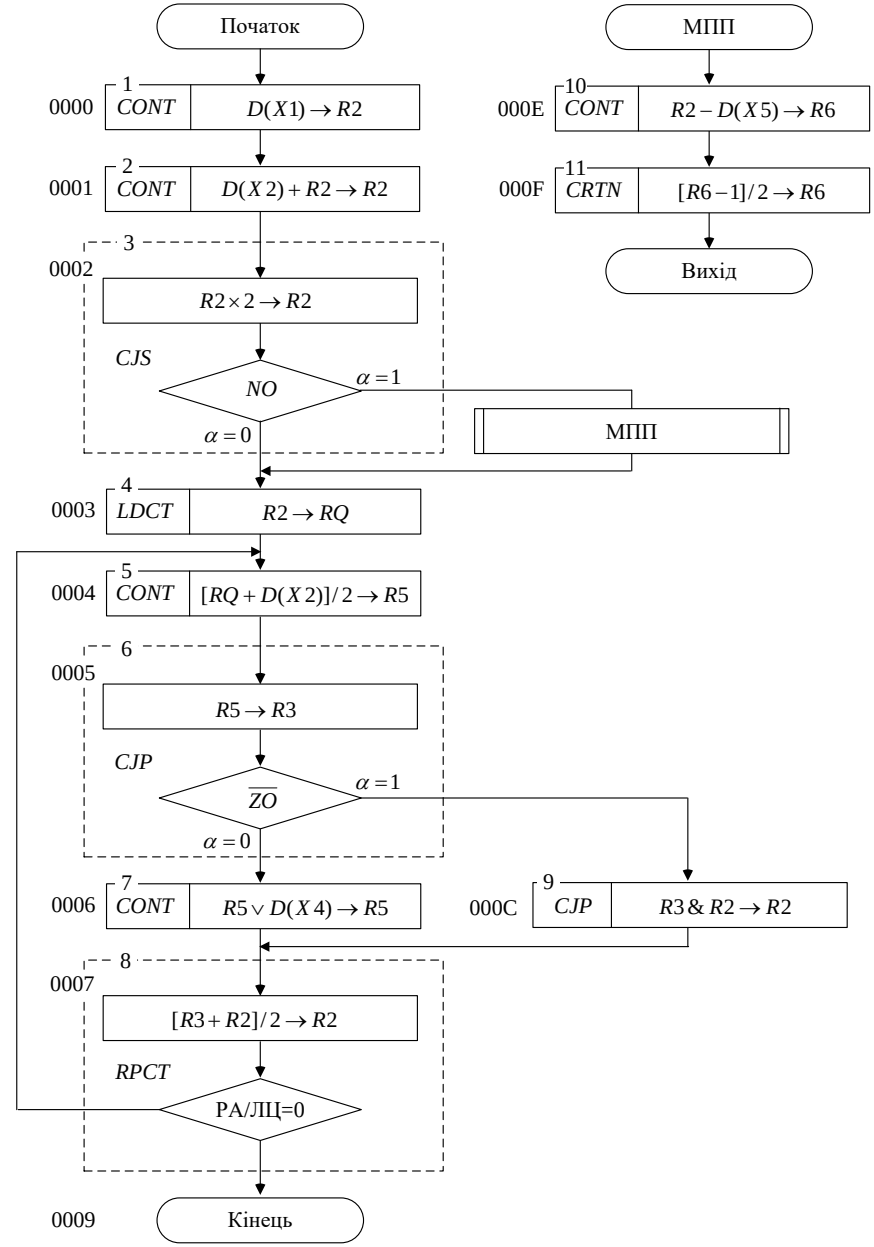


Рис. 2.33. Вихідний мікроалгоритм

```
     / мікропрограма із застосуванням символічного мікроасемблеру
     / область налагодження зв'язків
          link l2:ct;
     / область програми
0000      {or r2,r2,005h;}                    / X1(005h) → R2
0001      {add r2,r2,0c0h,z;}                 / X2(0C0h) + R2 → R2
0002 ml1  {or sl.16 r2,r2,z;cjs no,mpp;}      / R2×2 → R2 ; перехід на мікропідпрограму,
                                              / якщо результат від'ємний
0003 lab3 {add rq,r2,z;ldct 6;}               / R2 → RQ; завантаження в РА/ЛЦ кількості
                                              / циклів
0004      {add sr.2 r5,rq,055h,z;}            / (RQ + X3(0C0h)) / 2 → R5
0005      {add r3,r5,z,z;cjp not zo,lab1;}    / пересилання R5 → R3 ; умовний перехід на
                                              / мітку, якщо результат не дорівнює нулю
0006      {or r5,r5,0f00fh;}                  / R5 ∨ X4(0F00Fh) → R5
0007 lab2 {add sr.2 r2,r3,r2,z;rpct,lab3;}    /[R3 + R2]/ 2 → R2 ; інкремент вмісту РА/ЛЦ
                                              / та перевірка, якщо РА/ЛЦ не дорівнює нулю,
                                              / повернення на початок циклу
0009      {cjp nz, lend}                      / безумовний перехід на кінець мікропрограми
000C lab1 {and r2,r2,r3;cjp nz,lab2;}         / R3 & R2 → R2 , безумовний перехід на мітку
000E mpp  {sub r6,r2,0013h}                   / R2 − X5(0013h) → R6
000F      {sub r6,r6,z,nz;crtn;}              /[R6 − 1] / 2 → R6 , безумовний вихід із
                                              / мікропідпрограми
0010 lend {}                                  / кінець
```

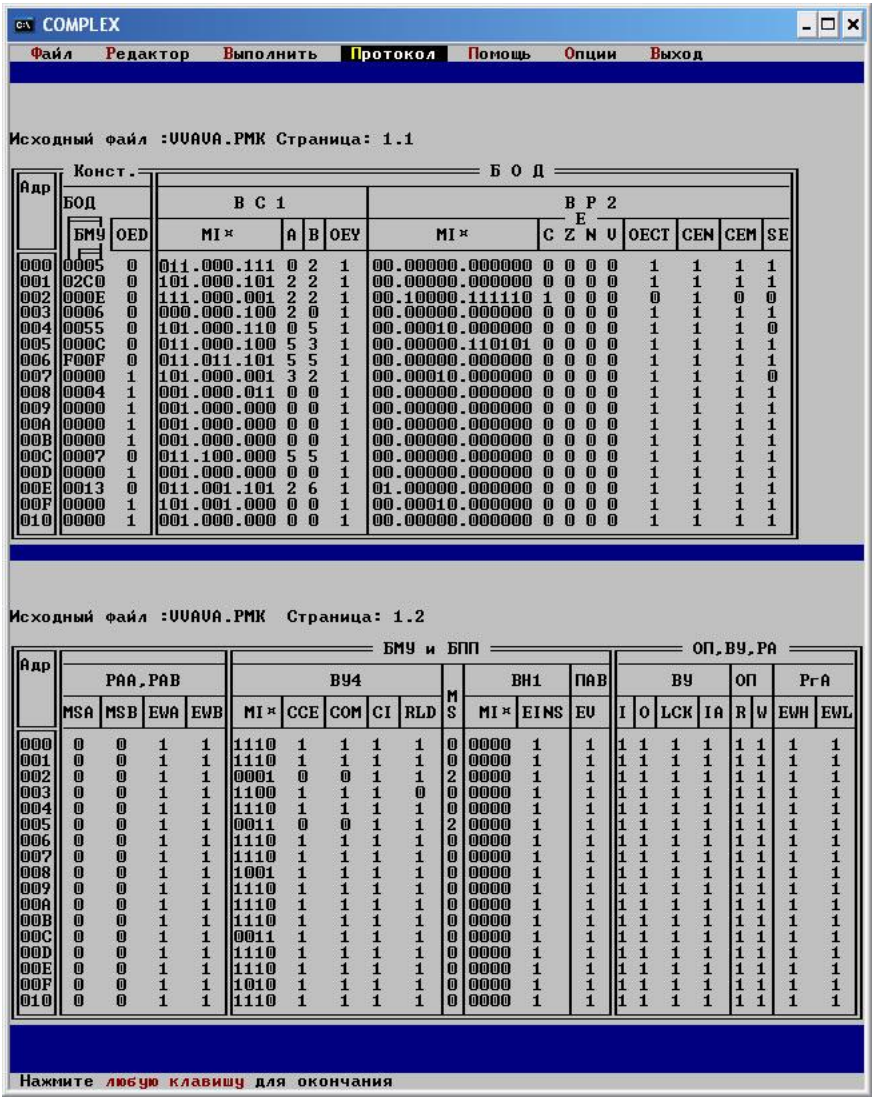


Рис. 2.34. Мікропрограма в машинних кодах

2.6. Блоки пріоритетних переривань

2.6.1. Структурна організація блоку пріоритетних переривань

До складу блоку пріоритетних переривань (БПП) входить схема векторних переривань (СВП) та перетворювач адреси (ПА). На СВП рис. 2.35 надходять запити на переривання $\overline{IRQ0}$, $\overline{IRQ1}$, …, $\overline{IRQ7}$ від восьми зовнішніх пристроїв, які фіксуються в восьмирозрядному регістрі запитів *IR*. Під час фіксації запитів на переривання може здійснюватися маскування запитів. Для цього в структурі СВП передбачено восьмирозрядний регістр маски *MR*, в який з ШД можна записати маску *MSC* або зчитати її на ШД.

Схема векторних переривань виділяє запит з максимальним пріоритетом серед немаскованих та порівнює пріоритет виділеного запиту з поточним пріоритетом, який зберігається в трирозрядному регістрі стану *SR*. Якщо пріоритет виділеного запиту дорівнює або перевищує поточний пріоритет, то СВП формує вектор переривання, який записується в регістр вектора *VR* та видається на вихід *VEC*, а також формує сигнал вимоги загального переривання *INT*.

Вектор переривання використовується для формування початкової адреси мікропрограми обслуговування переривання. Мікропрограми обслуговування переривань розміщено в ПМК. Сигнал *INT* надходить на МУ БМУ в якості логічної умови. Якщо *INT* = 1, то здійснюється перехід на мікропрограму обслуговування переривання.

Перетворювач адреси ПА, який є постійним запам‘ятовуючим пристроєм, містить вісім дванадцятирозрядних слів – початкових адрес мікропрограм обслуговування переривань. Кожному вектору переривання відповідає своя початкова адреса. Якщо $\overline{EV} = 0$, то початкова адреса з ПА надходить на ШД. Далі здійснити перехід на мікропрограму обслуговування переривання можна за допомогою мікрокоманди `{jmap;}`.

Схема векторних переривань є восьмирозрядною мікропрограмованою схемою, яка допускає нарощування.

2.6.2. Система мікрокоманд схеми векторних переривань

Для управління СВП в структурі мікрокоманди (рис. 2.36) передбачено чотирирозрядне поле СВП_МІ – мікроінструкція, та

поле $\overline{EINC}$ – дозвіл виконання мікроінструкції (*Enable INStruction*). Якщо $\overline{EINC} = 0$, то виконання мікроінструкції дозволено. СВП виконує 16 мікрокоманд, коди яких надходять з РМК на входи СВП_МІ (табл. 2.30). Їх можна поділити на чотири групи:

– мікрокоманди для роботи з регістром запитів на переривання *IR*:
– мікрокоманди для роботи з регістром маски:
– мікрокоманди для роботи з регістром стану:
– решта мікрокоманд:

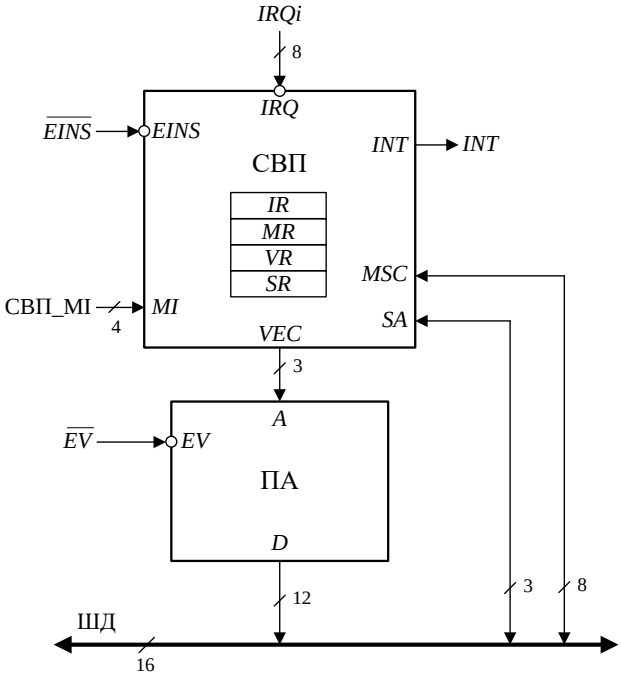


Рис. 2.35. Структура блока пріоритетних переривань

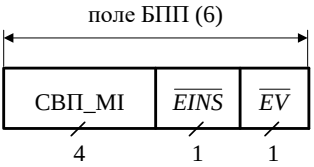


Рис. 2.36. Структура поля БПП мікрокоманди

Таблиця 2.30. Система мікрокоманд схеми векторних переривань

Мнемоніка мікрооперації	**Розряди поля СВП_МІ ($I_3..I_0$)**	**Призначення мікрокоманди**
1	2	3
Мікрокоманди для роботи з регістром запитів на переривання *IR*		
`{reset ir;}` – *загальна очистка IR*	`0000`	Мікрокоманда встановлює в «0» всі розряди регістра (всі запити на переривання знято).
`{clr ir, val;}` – *очистка окремих розрядів IR сигналами з шини маски*	`0010`	Мікрокоманда (*CLeaR*) встановлює в нуль ті розряди *IR*, яким відповідають одиниці в двійковому коді *val*. Значення *val* надходить з ШД на вхід *MSC* СВП. ***Наприклад:*** мікрокоманда `{clr ir,01100101%;}` скидає в нуль розряди: 0-й, 2-й, 5-й та 6-й регістра *IR*, тобто анулюються запити на переривання від ПВВ з пріоритетами 0, 2, 5, 6.
`{CLR IR,MR;}` – *очистка окремих розрядів IR сигналами з регістра маски*	`0011`	Мікрокоманда встановлює в нуль ті розряди *IR*, яким відповідають одиниці в регістрі маски. ***Наприклад:*** якщо *MR* = 10010010, то мікрокоманда `{clr ir,mr;}` встановлює в нуль 1-й, 4-й та 7-й розряди *IR* (анулюються запити на переривання від БПП з пріоритетами 1, 4, 7).

Продовження табл. 2.30

1	2	3
{CLR IR,VR;} *– очистка одного розряду IR під управлінням VR*	0100	Встановлює в нуль розряд *IR*, номер якого записано в *VR*. Ця мікрокоманда використовується для зняття запиту на переривання після його обслуговування. ***Наприклад:*** якщо *VR* = 011, то після виконання мікрокоманди`{clr ir,vr;}` буде встановлено в нуль 3-й розряд *IR* (знято запит на переривання від ПВВ з пріоритетом 3).
Мікрокоманди для роботи з регістром маски		
{READ MR;} *– читання з регістра маски*	0111	Мікрокоманда зчитує код маски з *MR* на ШД.
{RESET MR;} *очистка регістра маски*	1100	Мікрокоманда встановлює в нуль всі розряди *MR* (всі запити на переривання розмасковано, БПП здатний обробляти запити на переривання від будь-яких ПВВ).
{SET MR;} *встановлення в "1" всіх розрядів MR*	1000	Мікрокоманда записує одиниці у всі розряди регістра маски. Тобто всі запити на переривання від ЗП заборонено.
{CLR MR,val;} *очистка окремих розрядів регістра маски сигналами з шини маски*	1010	Мікрокоманда скидає в нуль розряди регістра маски, яким відповідають одиниці в двійковому коді *val*. ***Наприклад:*** мікрокоманда`{clr mr,01010000%;}` встановлює в нуль 4-й та 6-й розряди регістра маски.

Продовження табл. 2.30

1	2	3
{SET MR,val;} – *встановлення в одиницю окремих розрядів регістра маски сигналами з шини маски*	1011	Мікрокоманда встановлює в одиницю ті розряди *MR*, яким відповідають одиниці в двійковому коді *val*. Значення *val* надходить з ШД на вхід *MSC* БПП. ***Наприклад:*** мікрокоманда `{set mr,4;}` встановлює в одиницю 2-й розряд регістра маски.
{LOAD MR,val;} – *завантаження регістра маски значенням val*	1110	Мікрокоманда записує код маски (*val*), який встановлено на шині маски *MSC*, в регістр маски. ***Наприклад:*** мікрокоманда `{load mr,6;}` записує код маски 00000110 в регістр маски, внаслідок чого ігноруватимуться запити на переривання від ПВВ з пріоритетами 1 та 2.
Мікрокоманди для роботи з регістром стану		
{LOAD SR,val;} – *завантаження регістра стану значенням val*	1001	Мікрокоманда записує в *SR* код порога пріоритету (*val*), який надходить з ШД на вхід *SA* БПП. ***Наприклад:*** мікрокоманда `{load sr,2;}` означає, що дозволяється обслуговувати запити на переривання від ПВВ з пріоритетами 2 – 7, запити на переривання від ПВВ в пріоритетами 0, 1 будуть ігноруватися.

Продовження табл. 2.30

1	2	3
{READ SR;} – *читання слова з регістра стану*	0110	Мікрокоманда зчитує 3-розрядний код з *SR* на ШД.
Решта мікрокоманд		
{reset;} – *очистка IR*	0001	Мікрокоманда встановлює в нуль всі регістри схеми векторних переривань.
{READ VR;} – *читання вектора*	0101	Мікрокоманда забезпечує зчитування 3-розрядного вектора переривання *V* з регістра *VR* і надходження його на вихід *VEC* БПП, запис в регістр стану *SR* величини (*V*+1), та формування сигналу вимоги загального переривання *INT*.
{DI;} – *заборона запитів на переривання*	1101	Мікрокоманда (*Disable Interrupt*) забороняє фіксацію запитів на переривання від ПВВ у регістрі запитів на переривання СВП.
{EI;} – *дозвіл запитів на переривання*	1111	Мікрокоманда (*Enable Interrupt*) дозволяє фіксацію запитів на переривання від ПВВ у регістрі запитів на переривання СВП.

2.7. Проектування мікропроцесорних систем на секціонованому комплекті інтегральних схем

Комплект ІС серії К1804 дозволяє проектувати МПС, параметри яких можуть змінюватися у широких межах і мати довільну систему команд. Для побудови найпростішого процесора потрібні ІС трьох типів: К1804ВС1, К1804ВУ4 і ПЗП (наприклад, К556РТ5). Для побудови *n*-розрядного процесора потрібно $n/4$ процесорних елементів К1804ВС1. Блок мікропрограмного управління можна побудувати на одній ІС К1804ВУ4 і декількох ІС К556РТ5, що є постійними запам'ятовуючими пристроями об'ємом 512×8 біт, число ПЗП визначається довжиною слова мікрокоманди.

Параметри МПС багато в чому залежать від способу організації зв'язків між управляючою частиною та операційною частиною (ОЧ) процесора. Основними структурними елементами процесора є схема формування адреси мікрокоманд К1804ВУ4, пам'ять мікрокоманд (ПМК) і процесорні елементи К1804ВС1, з яких складається ОЧ процесора.

Розглянемо декілька різних структур процесорів, які відображають найбільш істотні зв'язки, що впливають на побудову мікропрограм і параметри процесора.

У процесорі на рис. 2.38, *а* застосовується регістр адреси мікрокоманди (РАМК). В цьому випадку виконання такту починається із завантаження адреси *A* поточної мікрокоманди в РАМК за позитивним перепадом синхросигналу *CLK*. Відповідно до адреси *A* з ПМК вибирається мікрокоманда *I*(*A*), в результаті виконання якої в ОЧ формуються ознаки *S*(*A*). Ці ознаки через мультиплексор МУ можуть поступати на вхід умови *CC* схеми ФАМ та враховуватися при формуванні наступної адреси $(A+1)$ наступної мікрокоманди. Найбільш тривалий шлях проходження сигналів показаний на рис. 2.38 пунктирною лінією. Максимальна тривалість такту *T*, як випливає з діаграми на рис.2.37, дорівнює сумарній затримці сигналів у всіх блоках. Суміщати в часі виконання однієї мікрокоманди з вибіркою із ПМК іншої в даному випадку є неможливим. Такий спосіб виконання мікрокоманд називають послідовним.

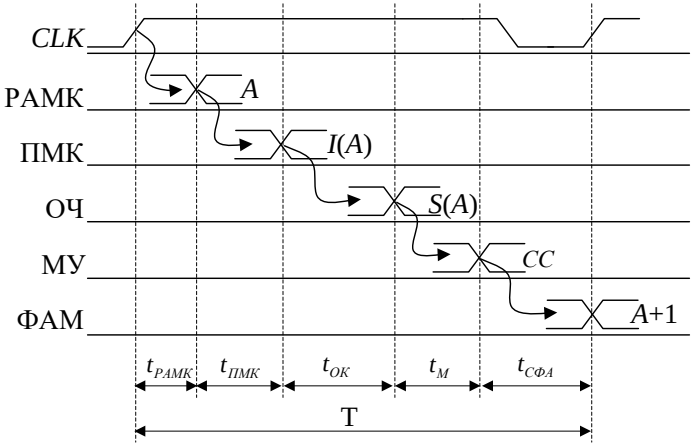


Рис. 2.37. Часова діаграма

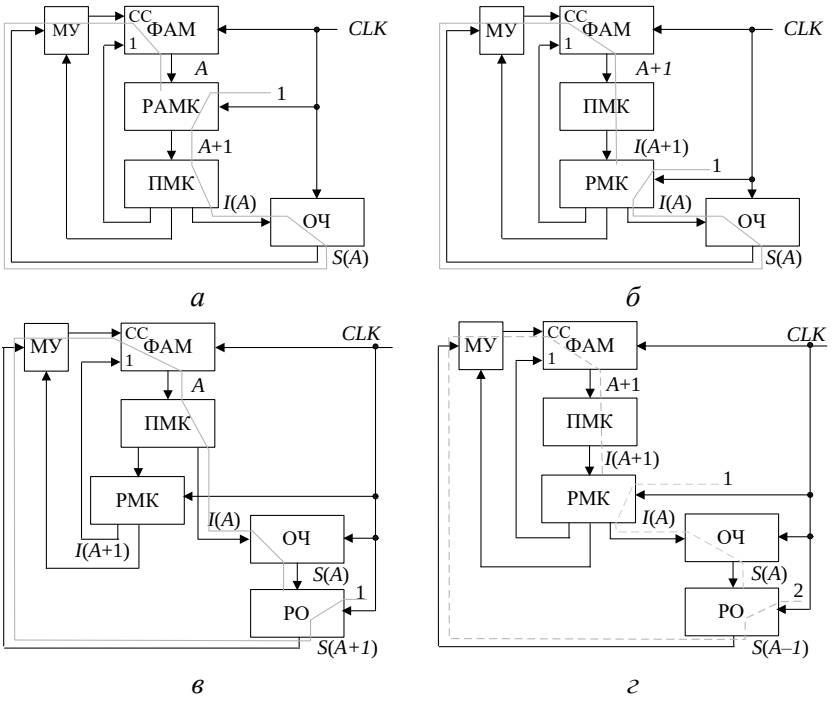
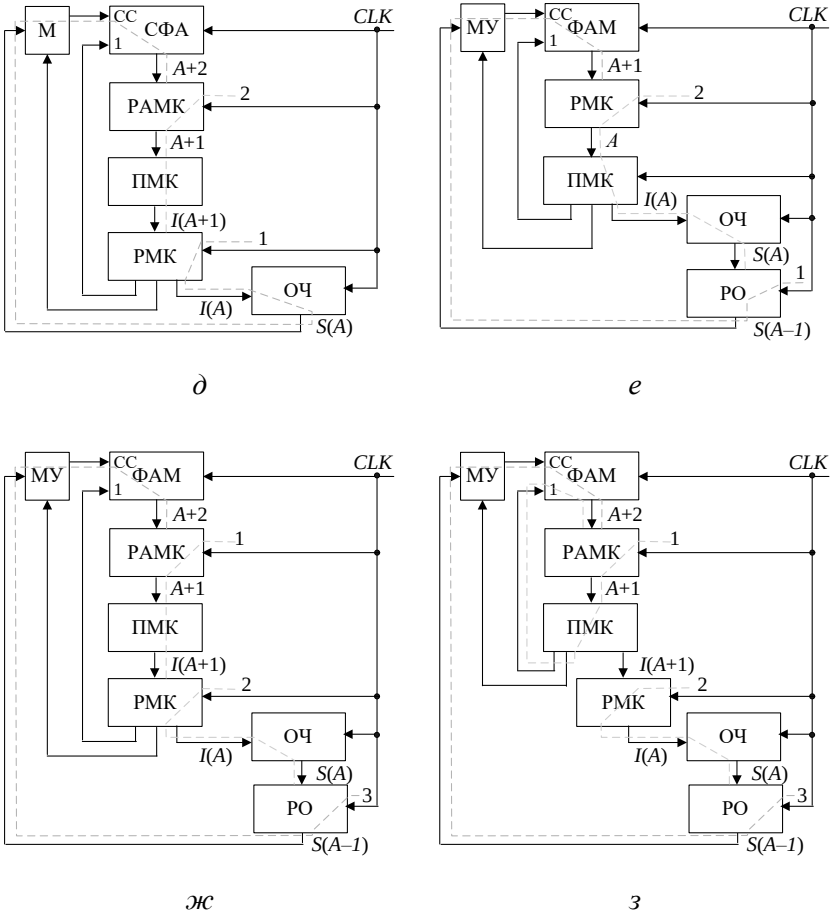


Рис. 2.38. Приклади структур процесора



Продовження рис. 2.38

У схемі на рис. 2.38, *б* використовується регістр мікрокоманд (РМК). В цьому випадку такт починається з запису мікрокоманди *I*(*A*) в РМК, а завершується зчитуванням із ПМК наступної мікрокоманди *I*(*A*+1).

Зменшити довжину РМК дозволяє схема, що наведена на рис. 2.38, *в*. У регістрі ознак (РО) запам'ятовуються ознаки, отримані в ОЧ. Число ознак менше за число управляючих входів ОЧ. Регістр РМК містить тільки розряди управління МУ і ФАМ.

У процесорах на рис. 2.38, *б* і 2.38, *в*, як і в процесорі на рис. 2.38, *а*, не можна поєднувати в часі виконання однієї мікрокоманди з вибіркою іншої.

Комбінуючи три базові способи організації структур, можна добитися зменшення тривалості такту за рахунок поєднання в часі деяких етапів виконання мікрокоманд. Такий спосіб виконання мікрокоманд називається конвеєрним. Приклади конвеєрної реалізації показані на рис. 2.38, *г* – рис.2.38, *з*.

Процесори на рис. 2.38, *г* – рис.2.38, *е* містять по два регістри.

Циклічний шлях проходження сигналу в схемах розділений на два незалежні шляхи. Наприклад, на 2.38, *г* один шлях містить РМК і ОЧ, а інший – РО, МУ, ФАМ і ПМК. При цьому виконання поточної мікрокоманди в ОЧ поєднується з вибіркою з ПМК наступної мікрокоманди.

Тривалість такту в цьому випадку менше, ніж при послідовному виконанні мікрокоманд. Схеми на рис. 2.38, *е* – рис.2.38, *ж* містять по три регістри і, відповідно, по три незалежні шляхи проходження сигналів.

В одному такті на різних стадіях обробки знаходяться три мікрокоманди. Тривалість такту при цьому зменшується.

Зауважимо, що при реалізації умовних переходів, коли вибірка адреси наступної мікрокоманди залежить від результату виконання попередньої, поєднання в часі виконання етапів різних мікрокоманд порушується. Умовний перехід може бути виконаний тільки в тому такті, коли необхідна інформація знаходитиметься в РО. Це призводить до збільшення числа тактів при виконанні мікропрограм.

Таким чином, розглянуті структури відрізняються тривалістю такту, кількістю тактів при реалізації заданого алгоритму і апаратурними витратами.

Зменшення тривалості такту за рахунок конвеєрної обробки мікрокоманд супроводжується зростанням апаратурних витрат і, як правило, збільшенням кількості тактів виконання мікропрограми. І з цього випливає, що вибір структури процесора повинен здійснюватися з врахуванням алгоритмів, що реалізовуються.

За однакового способу обробки мікрокоманд процесори можуть мати схемні відмінності, обумовлені системою команд.

Розглянемо структуру процесора, що реалізовуватиме певну скорочену систему команд.

Модель програміста процесора для реалізації заданої системи команд показана у табл. 2.31.

Таблиця 2.31. Модель програміста

Номер регістру	**Призначення регістрів**	
*R*0		**Регістри загального призначення (РЗП)** Застосовуються для адресації операндів у двоадресних командах. Доступ на рівні програм.
*R*1		
*R*2		
*R*3		
*R*4		
*R*5		
*R*6	Покажчик стека (ПС)	
*R*7	Лічильник команд (ЛК)	
*R*8	Регістр команд (РК)	**Робочі регістри** Застосовуються для виконання обчислень на мікропрограмному рівні, зберігання слова команди на протязі часу її виконання
*R*9	Регістр маски (РМ)	
*R*10		
*R*11		
*R*12		
*R*13	Покажчик адреси (ПА)	
*R*14		
*R*15	Регістр акумулятор (РА)	

До складу процесора входять шістнадцять регістрів $R15-R0$, які створюють надоперативний запам'ятовуючий пристрій (НОЗП). Регістри $R7-R0$ належать до регістрів загального призначення, за їх допомогою реалізуються різні способи адресації операндів, РЗП можуть використовуватися як приймачі так і джерела операндів. Доступ до цих регістрів може бути здійснений на рівні програми. Регістр $R6$ використовується як покажчик стека, $R7$ виконує функції лічильника команд (ЛК).

Регістри $R15-R8$ є робочими регістрами (*Rr*), які застосовуються для зберігання слова команди та операндів на

протязі всіх етапів її виконання та проміжних даних та результатів. Доступ до робочих регістрів може бути здійснений тільки на мікропрограмному рівні.

У функціональному відношенні всі команди можна класифікувати наступним чином:

– основні команди,
– команди передачі управління,
– системні команди.

До основних належать команди пересилки, зсуву, арифметичних та логічних перетворювань інформації. Команди передачі управління забезпечують розгалуження програми. Це безумовний і умовні переходи, команди умовного та безумовного переходу до підпрограмам і повернення з них. Системні команди задають режими роботи процесора, управляють перериваннями програми, тощо.

Система команд включає безадресні, одноадресні і двоадресні команди. Формати основних одноадресних і двоадресних команд показані на рис. 2.39, *а, б*. Адреси РЗП наведені в табл. 2.32, а приклад кодування способів адресації операндів для команд запропонованого формату – в табл. 2.33.

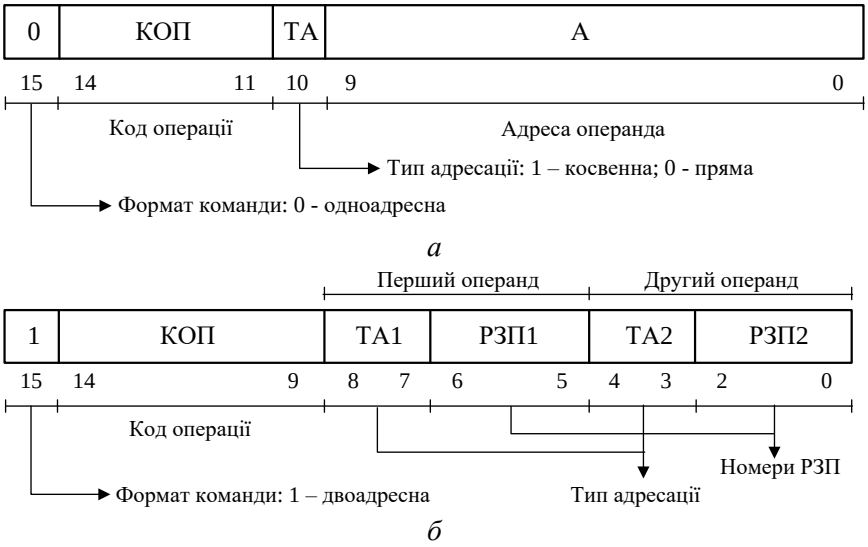


Рис. 2.39. Формат команди: а – одноадресної; б – двоадресної.

Таблиця 2.32. Адреси РЗП у НОЗП

РЗП	**Адреса в НОЗП**		
*R*0	0	0	0
*R*1	0	0	1
*R*2	0	1	0
*R*3	0	1	1
*R*4	1	0	0
*R*5	1	0	1
*R*6	1	1	0
*R*7	1	1	1

Таблиця 2.33. Кодування типів адресації

Код типу адресації		**Назва типу регістрової адресації**
0	0	Пряма регістрова
0	1	Непряма регістрова
1	0	Автоінкрементна
1	1	Автодекрементна

Взагалі можуть бути реалізовані наступні типи адресації операндів:

Без застосування РЗП (одноадресні команди основної групи, команди передачі управління, команди вводу-виводу):

– пряма,
– непряма.

Із застосуванням РЗП (команди основної групи):

– пряма,
– непряма,
– автоінкрементна,
– автодекрементна,
– непряма автоінкрементна,
– непряма автодекрементна,
– індексна,
– непряма індексна.

Схеми формування виконавчих адрес операндів при різних типах регістрової адресації показані на рис. 2.40.

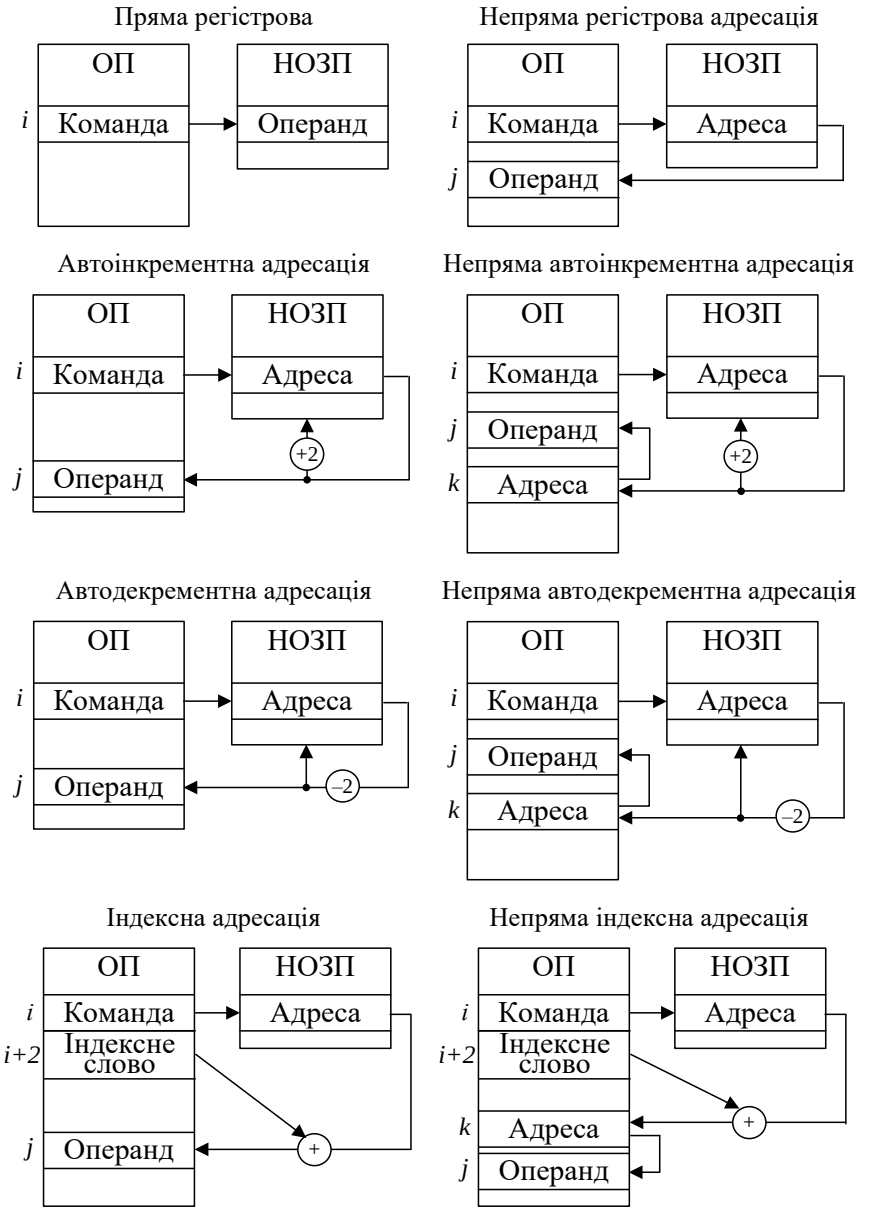


Рис. 2.40. Схеми формування виконавчих адрес операндів

Співвідношення між значеннями *i*, *k* і *j* адрес слів і загальної пам'яті можуть бути різними. При використанні способів автоінкрементної та автодекрементної адресації (прямої і непрямої) вміст регістру РЗП, що виконує функції покажчика адреси в НОЗП, змінюється на два, якщо операндом є слово, або на одиницю, якщо операндом є байт.

Розглянемо приклади формування виконавчих адрес операндів за застосування різних способів адресації із використанням РЗП. Припустимо, що адреса регістра $R5$ – 101, в якому знаходиться операнд, вказана в молодшій тріаді коду команди (рис. 2.39). У розглянутих прикладах виконується команда інкременту.

Пряма регістрова адресація. В регістрі $R5$ знаходиться безпосередньо операнд. До вмісту $R5$ додається 1.

Непряма регістрова адресація. Вміст комірки загальної пам'яті з адресою $0010h$, яка записана в регістрі $R5$, збільшується на одиницю.

Автоінкрементна адресація. Вміст комірки загальної пам'яті з адресою $0022h$, яка зберігається в регістрі $R5$, збільшується на одиницю. Після чого адреса в регістрі $R5$ збільшується на два.

Непряма автоінкрементна адресація. Вміст комірки пам'яті, адреса якої $0100h$ записана в $R5$, є адресою операнду ($0132h$). Операнд розташований в пам'яті за адресою $0132h$ збільшується на одиницю.

Автодекрементна адресація. Вміст $R5$ зменшується на два. Нове значення $R5$ ($j = 0770h$) використовується як адреса операнда. Операнд розташований у пам'яті за адресою $0770h$ збільшується на одиницю.

Непряма автодекрементна адресація. Вміст $R5$ зменшується на два і використовується як адреса комірки, в якій знаходиться адреса операнда ($j = 1112h$). Операнд, записаний в загальній пам'яті за адресою $1112h$, збільшується на одиницю.

Індексна адресація. Адреса операнда визначається додаванням вмісту $R5$ та індексного слова, яке записане в загальній пам'яті безпосередньо за словом команди (команда зберігається за адресою $i = 0100h$, а індексне слово – за адресою $(i+2) = 0102h$). Операнд у комірці пам'яті $1203h$ збільшується на одиницю.

Непряма індексна адресація. Додаванням вмісту регістру $R5$ та індексного слова формується адреса комірки, в якій зберігається

адреса операнда. Операнд за адресою 0400*h* збільшується на одиницю.

Для адресації у команді може бути використаний будь-який РЗП, в тому числі і лічильник команд (регістр $R7$). Адресація з використанням лічильника команд дає можливість побудувати програму, яка не втрачає працездатність при переміщенні її в будь-яку область пам'яті. Способи адресації, що відповідають автоінкрементній, непрямій автоінкрементній, індексній та непрямій індексній, з використанням лічильника команд отримали назви безпосередня, абсолютна, відносна і непряма відносна відповідно.

У двоадресних командах типи адресації і поле адреси джерела операнда і приймача операнда можуть бути різними.

Виконання основних команд основної групи включає етапи:

1. Вибірка команди;
2. Розпакування команди
   - визначення формату команди та типу адресації;
   - обчислення виконавчих адрес і вибірка операндів;
3. Виконання операції і розміщення результату;
4. Формування адреси наступної команди.

Команда вибирається із загальної пам'яті за адресою, що знаходиться в ЛК і завантажується в регістр команд. Відповідно до типу адресації формуються виконавчі адреси операндів, здійснюється задане кодом операції перетворення інформації і засилання результату за адресою приймача. Адреса наступної команди формується в лічильнику команд збільшенням на два його вмісту.

Команди передачі управління та вводу-виводу належать до групи одноадресних команд. На етапі виконання команди умовного переходу перевіряється умова. Якщо умова виконується, то в ЛК формується нова адреса переходу. Якщо ж умова не виконується, то природний порядок виконання програми не порушується, тобто вміст ЛК збільшується на два і виконується наступна команда.

Розглянемо структуру МПС із процесором, що складається з блоку мікропрограмного управління (розділ 2.5), постійного запам’ятовуючого пристрою, шістнадцятирозрядного блоку обробки даних, побудованого на чотирьох чотирирозрядних процесорних елементах (розділ 2.2, розділ 2.4) та схеми управління станами та зсувами (розділ 2.3). Загальна структура МПС зображена на рис. 2.41.

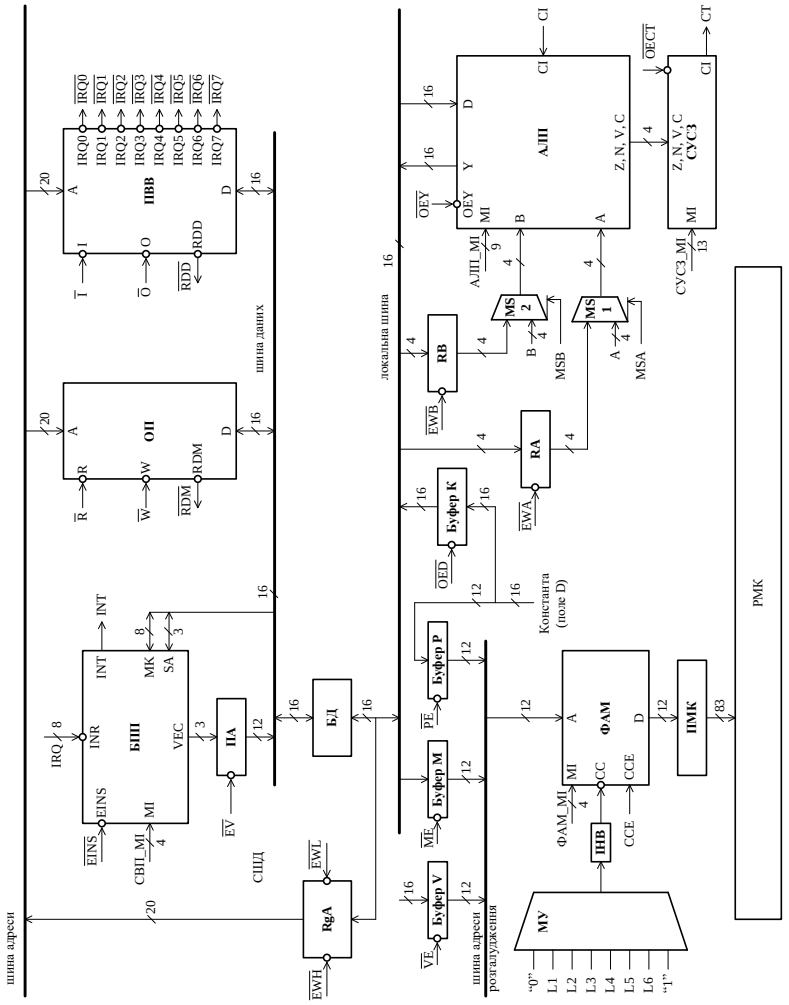


Рис. 2.41. Загальна структура мікропроцесорної системи

Обмін даними з основною пам'яттю (ОП) і зовнішніми пристроями (ЗП) здійснюється у програмному режимі по двоспрямованій шині даних (ШД). Операції обміну інформацією з ОП та ЗП супроводжуються сигналами запису/читання *W* (*Wright*) /*R* (*Right*) та вводу/виводу *I* (*Input*) /*O* (*Output*) відповідно. Всі управляючи сигнали кодуються у відповідних полях мікрокоманди, що формується в БМУ. Перед записом на шині адреси ША виставляється адреса комірки пам'яті або порту ЗП. На ШД виставляються дані, призначені для виводу. В якості регістра адреси та регістра даних застосовуються зовнішній регістр *RgA* та буфер даних БД. Інформація, що вводиться з ОП або ЗП приймається у процесор по ШД: спочатку на ША виставляється адреса джерела даних, після чого за відповідним управляючим сигналом дані виставляються на ШД та фіксуються в процесорі. ОП та ЗП сигналізують процесору про завершення операції фіксації даних у пристрої під час виводу даних, або готовності даних на ШД під час вводу даних за допомогою сигналів готовності *RDM* (від ОП) та *RDD* (від ЗП), що поступає для аналізу на мультиплексор умов МУ.

У склад процесору входить НОЗП, що складається з шістнадцяти шістнадцятирозрядних регістрів. Чергова команда, що вибирається з ОП на першому етапі виконання розміщується в один з робочих регістрів, відповідно моделі програміста (табл. 2.31) це регістр *R*8. За допомогою маски, що зберігається у регістрі *R*9 можна відокремити деякі розряди слова команди та виконати їх аналіз у БМУ. Так виконується аналіз розрядів ФК, ТА і КОП. Під час аналізу поля КОП розряди цього поля видаються на локальну шину (ЛШ), відповідні розряди якої приєднані до входів Буферу *M* (Додаток А, рис. А.3). Під дією управляючих сигналів КОП записується в Буфер *M*, де формується адреса переходу на мікропідпрограму виконання команди у ПМК.

Для адресації регістрів загального призначення через адресні поля команди в процесі формування виконавчих адрес застосовуються зовнішні регістри *PA* та *PB*. Інформація з адресних полів слова команди по ЛШ передається у зазначені регістри. Мультиплексори *MA* та *MB* дозволяють адресувати регістри НОЗП, як через поля команди так і через поля мікрокоманди у ПМК.

З ціллю зсувів слів, обробки ознак та формування вхідного переносу застосовується СУСЗ.

Буфер *V* БМУ застосовується для реалізації переривань. Під час виникнення переривання на входи Буферу *V* подаються вектори переривань, що є адресами мікропідпрограм обробки переривань.

Шина даних процесора має довжину 16 розрядів. Процесор може здійснювати обмін шістнадцятирозрядними словами.

Шина адреси має довжину 20 розрядів. Ємність ОП становить 1М 16-розрядних слів. Мінімальна інформаційна одиниця доступу – 16-розрядне слово, доступ до окремого байта слова неможливий. Слова в ОП адресуються тільки парними адресами.

Обмін інформацією із ЗП здійснюється за допомогою спеціальних команд вводу-виводу *IN* (*Input*) *OUT* (*Output*). Адреси регістрів ЗП (портів вводу-виводу: РС – регістру стану та РД – регістру даних) утворюють з комірками ОП єдиний адресний простір.

Під час розробки мікропрограм на першому етапі складається функціональний мікроалгоритм (Ф-мікроалгоритм) перетворення інформації, що включає узагальнені мікрооперації. Слід зазначити, що такі мікрооперації можуть бути відсутні у системі мікрооперацій застосованого ПЕ. Узагальнені мікрооперації записують змістовними термінами або операторами присвоєння. На підставі Ф-мікроалгоритму розробляється функціонально-структурний мікроалгоритм (ФС-мікроалгоритм), що враховує реальну систему мікрооперації та зв’язки, власні обраній структурі процесора. В операційних вершинах ФС-мікроалгоритму розміщуються тільки ті мікрооперації, що виконуються за один такт роботи процесора. ФС-мікроалгоритм є вихідною інформацією для розробки мікропрограм та розміщення їх у ПМК.

Розглянемо приклад розробки мікроалгоритмів для основних етапів виконання команди. Припустимо, що процесор реалізує скорочену систему команд, наведену у табл. 2.34.

Ф-мікроалгоритм основних етапів виконання команди наведений на рис. 2.42. На етапі вибору команда зчитується з ОП за адресою, що надійшла із лічильника команд у регістр РА. Далі процесор генерує сигнал читання *R* і переходить у стан очікування сигналу готовності. Вибране слово команди поступає на ШД, ОП

сигналізує процесору про готовність даних сигналом *RDM* і команда приймається у РК НОЗП.

Таблиця 2.34. Система команд процесора

Двоадресні команди														
Мнемоніка	**КОП**	**Двійковий код команди у ОП**									**Адреса МПП у ПМК**			
		ФК	КОП				ТА1	А1	ТА2	А2		КОП		
ADD	00010	1	0	0	0	0 1	* *	* * * *	* *	* * * *	00	00010		00000
SUB	00100	1	0	0	1	0 1	* *	* * * *	* *	* * * *	00	00100		00000
MUL	00110	1	0	0	1	1 0	* *	* * * *	* *	* * * *	00	00110		00000
DIV	01110	1	0	0	1	1 1	* *	* * * *	* *	* * * *	00	01110		00000
Одноадресні команди														
Мнемоніка	**КОП**	**Двійковий код команди у ОП**									**Адреса МПП у ПМК**			
		ФК	КОП				ТА	А				КОП		
MOV	0011	0	0	0	1	1	*	* * * * * * * * * *			00	0011		000000
JMP	1001	0	1	0	0	1	*	* * * * * * * * * *			00	1001		000000
CJS	1000	0	1	0	0	0	*	* * * * * * * * * *			00	1000		000000
IN	1100	0	1	1	0	0	*	* * * * * * * * * *			00	1100		000000
OUT	1101	0	1	1	0	1	*	* * * * * * * * * *			00	1101		000000

На етапі розпакування команди визначається її формат та тип адресації операндів. Залежно від формату команди виконується розпакування одноадресної, або двоадресної команди. Залежно від типу адресації обчислюється виконавча адреса операндів і вибір за цією адресою операндів у робочі регістри НОЗП. За двоадресної адресації один з операндів, що адресується полем команди РЗП1 пересилається в робочий регістр НОЗП $Rr1$, а другий, що адресується через адресне поле РЗП2, – в робочий регістр $Rr2$. Для одноадресних команд в робочий регістр пересилається тільки один операнд, що адресується адресним полем А (рис. 2.42).

Приклад обчислення виконавчої адреси *A* і вибірки операнда у робочий регістр НОЗП для непрямої адресації показаний на

рис. 2.43, де через РЗП*i* позначений номер регістру, що вміщує адресну інформацію.

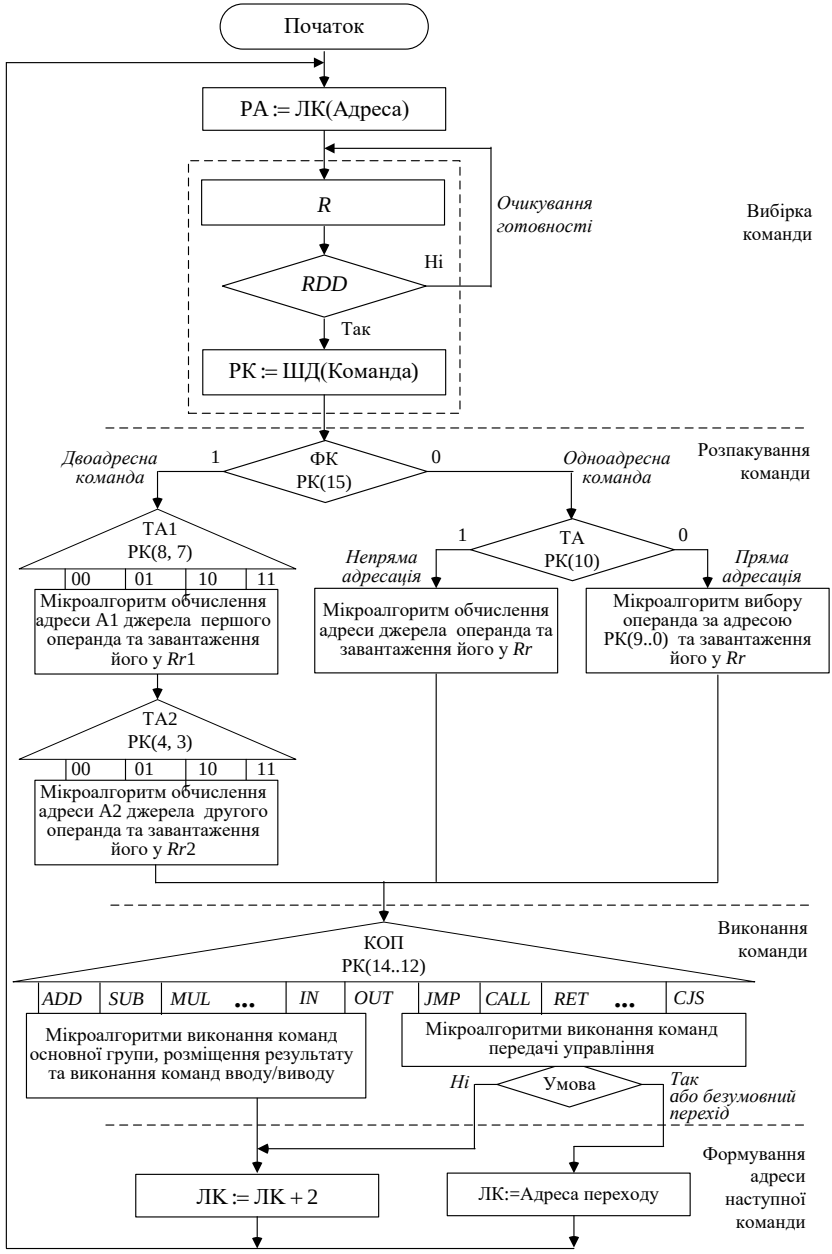


Рис. 2.42. Ф-мікроалгоритм виконання команди

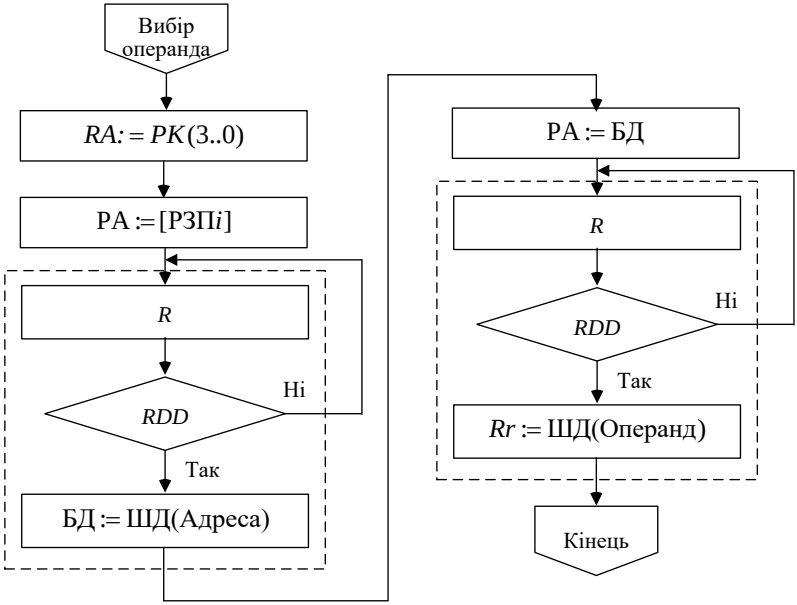


Рис. 2.43. Ф-мікроалгоритм вибірки операнда

На етапі виконання команди здійснюється перехід на мікроалгоритм виконання заданої операції відповідно коду операції. При виконанні команд передачі управління здійснюється формування адреси переходу в ЛК. При виконанні одноадресних і двоадресних команд основної групи здійснюється задане перетворення інформації. Результат формується в одному з робочих регістрів *Rr* , з якого можна видати результат на ШД для запису в пам'ять.

Під час прямої регістрової адресації результат записується НОЗП і застосовується у подальших обчисленнях, або за допомогою команди пересилки *MOV*, засилається в ОП. При інших способах адресації результат може бути записаний в ОП за адресою другого операнда, яка на етапі розпакування команди була збережена в РА. Слід зазначити, що мікроалгоритми виконання команд складаються проектувальниками в залежності від цільової функції МПС.

Формування адреси наступної команди за природного порядку виконання команд зводиться до збільшення на два вмісту ЛК.

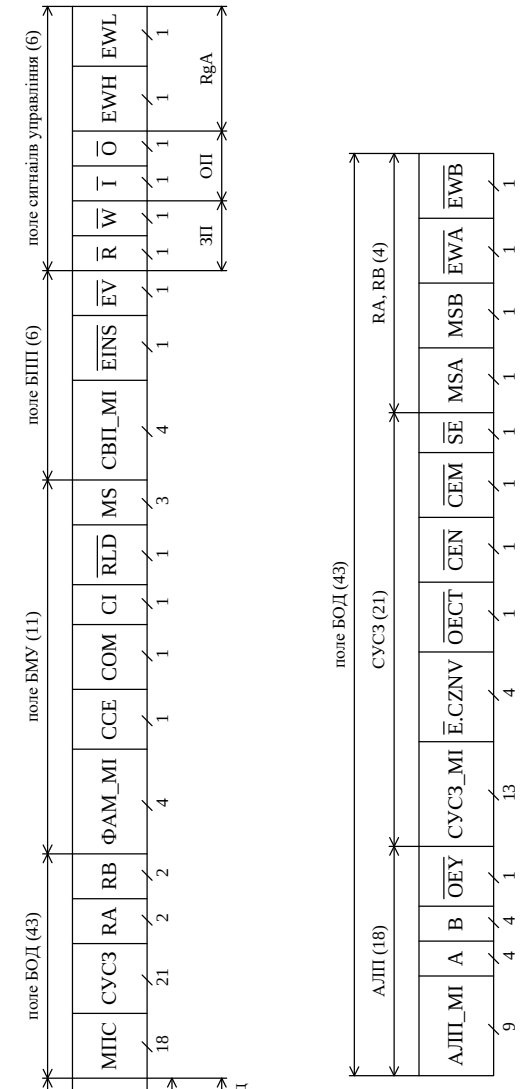


Рис. 2.44. Мікрокоманда для мікропроцесорної системи: *а* – структура мікрокоманди; *б* – поля мікрокоманди для управління БОД

Мікрокоманда для ЕОМ на рис. 2.44 має наступну структуру:

Поле констант:

OED – дозвіл видачі константи;
D – дані - константа, що видається на ШД;

Поле управління БОД:

АЛП_МІ – мікроінструкція;
A – адреса регістра НОЗП за каналом *A*:;
B – адреса регістра НОЗП за каналом *B;*
OEY – дозвіл видачі результату на ШД:

Поле управління СУСЗ:

СУСЗ_МІ – мікроінструкція;
EZ – дозвіл запису ознаки *Z*;
EN – дозвіл запису ознаки *N*;
EC – дозвіл запису ознаки *C*;
EV – дозвіл запису ознаки *V*;
OECT – дозвіл видачі умови;
CEM – дозвіл запису всіх ознак в *RM*;
CEN – дозвіл запису всіх ознак в *RM*;
SE – дозвіл зсуву;

Поле управління БМУ:

ФАМ_МІ – мікроінструкція;
CCE – дозвіл аналізу умов;
COM – інвертування умови;
CI – вхідне перенесення для схеми інкременту;
RLD – дозвіл запису в регістр адреси/лічильника;
MS – управління мультиплексором умов;

Поле управління регістрами і мультиплексорами:

MSA – управління мультиплексором *MA*;
MSB – управління мультиплексором *MB*;
EWA – дозвіл запису в *PAA*;
EWB – дозвіл запису в *PAB*;
EWH – дозвіл запису в старші розряди *PA*;
EWL – дозвіл запису в молодші розряди *PA*;

Поле управління ОП і ВУ:

R – сигнал читання даних з ОП;
W – сигнал запису даних в ОП;

I – сигнал вводу даних з ЗП
O – сигнал виводу даних на ЗП

При розробці ФС-мікроалгоритмів узагальнені мікрооперації у Ф-мікроалгоритмі замінюють на мікрооперації, що входять до складу системи мікрооперацій ЕОМ.

Розглянемо фрагменти ФС-мікроалгоритмів, що забезпечують наступні етапи виконання команди – вибірка команди з пам'яті (рис. 2.45), розпакування команди (рис. 2.46) та перехід на мікропідпрограму виконання операції (рис. 2.47), а мікропрограма в кодах мікрокоманд, що відповідає наданим фрагментам ФС-мікроалгоритмів приведена на рис. 2.49.

Мікропрограма розміщується в ПМК розпочинаючи з нульової адреси. Адреси мікрокоманд представлені в шістнадцятирозрядній системі числення. Активний рівень сигналу готовності пам'яті (*RDM*) є нульовим. Функціональне застосування регістрів НОЗП відповідає моделі програміста поданій у табл. 2.31.

Фрагмент ФС-мікроалгоритму, що забезпечує розпаковку відповідних форматів команд, представлений на рис. 2.46. Розряд, що визначає за формат команди у слові команди співпадає зі знаковим розрядом, тому визначити формат команди можна аналізуючи знак слова команди. Під час визначення типу адресації також здійснюється аналіз відповідних розрядів слова команди (табл. 2.34). Аналізується розряди, виділені за допомогою масок (маски завантажуються в регістр *R*9). Маски зазвичай добираються наступним чином: в розряд маски відповідний розряду слова команди, що аналізується, записують одиницю, інші розряди дорівнюють нулю. Між регістром маски та регістром команди виконують мікрооперацію логічного множення (кон'юнкція). При цьому результат виконання мікрооперації в АЛБ нікуди не записується – приймач *NIL*). В ІС ВС1 під час проходження інформації через АЛБ формується ознака рівності результату нулю *ZO*. За рівності нулю результату виконання мікрооперації слід зробити вивід, що аналізований розряд слова команди дорівнює нулю, та навпаки. На рис. 2.46 зображені гілки мікроалгоритму відповідні розпакуванню одноадресної команди і прямої адресації операндів.

Галуження на декілька напрямів при розробці ФС-мікроалгоритмів зручно здійснювати з використанням

мікроінструкції переходу *JMAP* блоку мікропрограмного управління. В цьому випадку перехід здійснюється за один такт, а адреса переходу визначається інформацією у Буфері *M*. Для аналізу поля КОП слова команди до входів Буферу *M* можна підключити відповідні розряди ЛШ, що забезпечить аналіз коду операції для будь-якої команди. За такого способу галуження адреси переходів на мікропідпрограми виконання операцій матимуть вигляд, показаний в табл. 2.34.

Фрагмент ФС-мікроалгоритму переходу на виконання одноадресної команди за кодом операції показаний на рис. 2.47. Перед виконанням функції *JMAP* в над вмістом РК виконується мікрооперація, наприклад, додавання нуля, результат нікуди не зберігається, але за встановлення сигналу $\overline{OEY}$ видається на ШД, звідки поступає у Буфер *M*.

Мікропрограми, розроблені із застосуванням символічного мікроасемблеру та у кодах мікрокоманд, що реалізують мікроалгоритми на рис. 2.45 – рис. 2.47 приведені на рис. 2.48, рис. 2.49 відповідно.

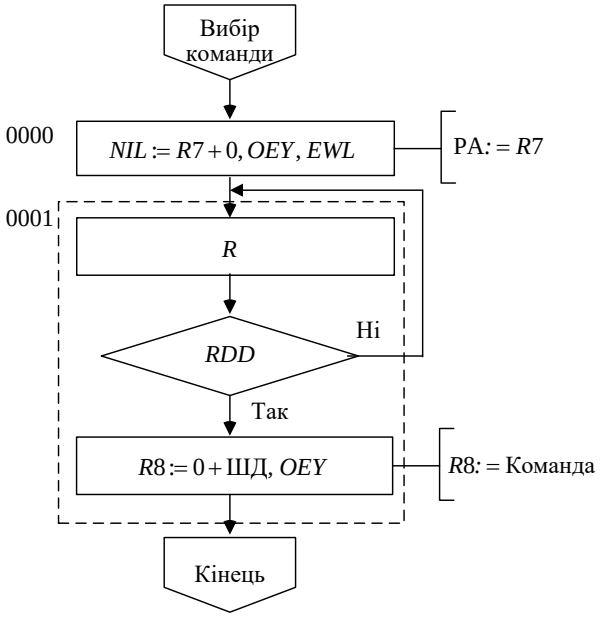


Рис. 2.45. ФС-мікроалгоритм вибірки команди з ОП

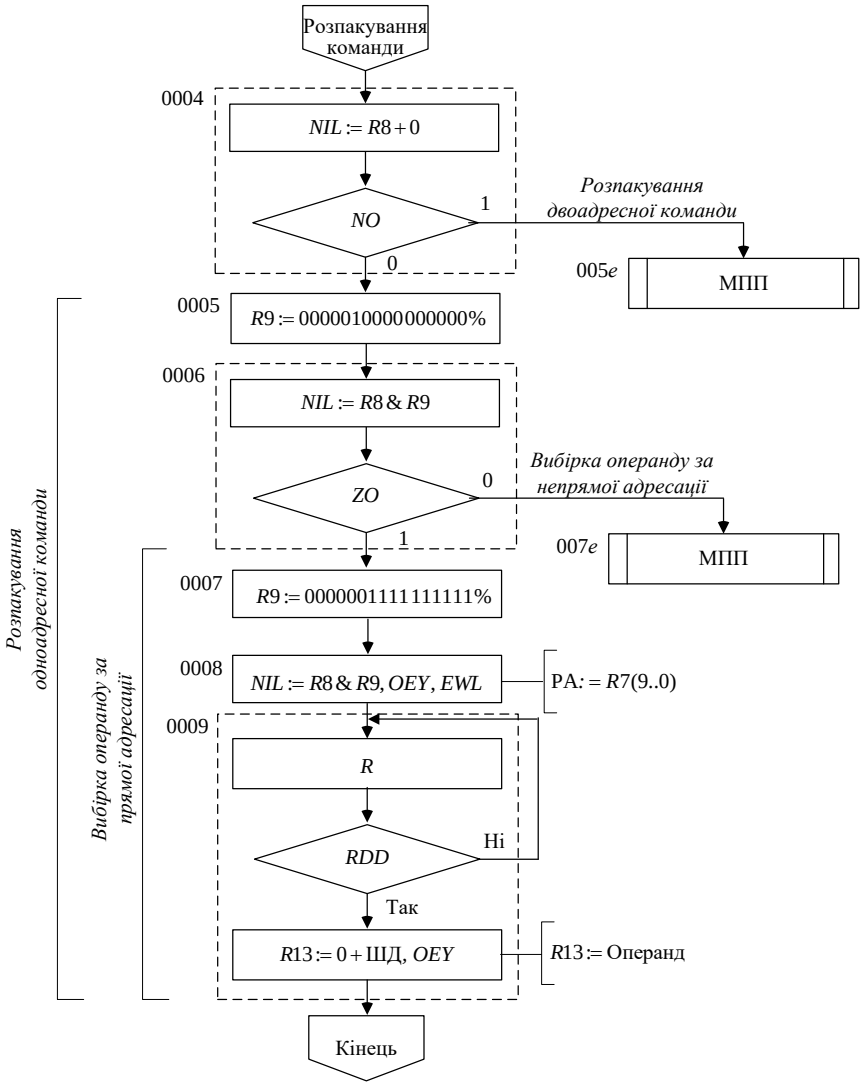


Рис. 2.46. ФС-мікроалгоритм розпакування одноадресної команди

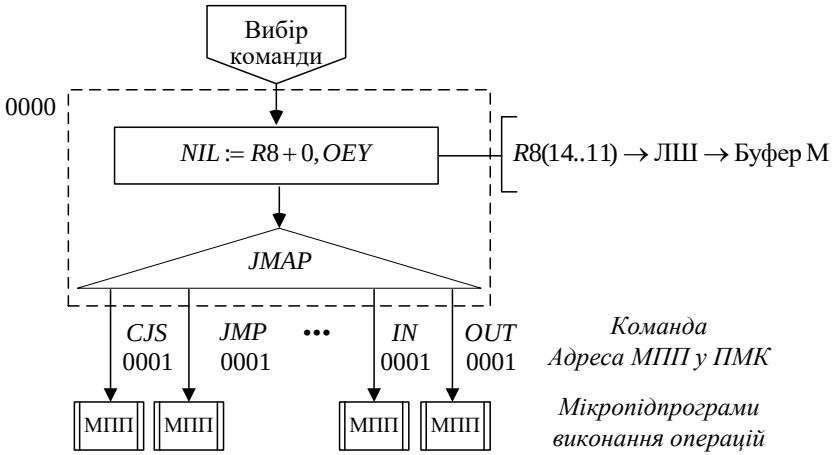
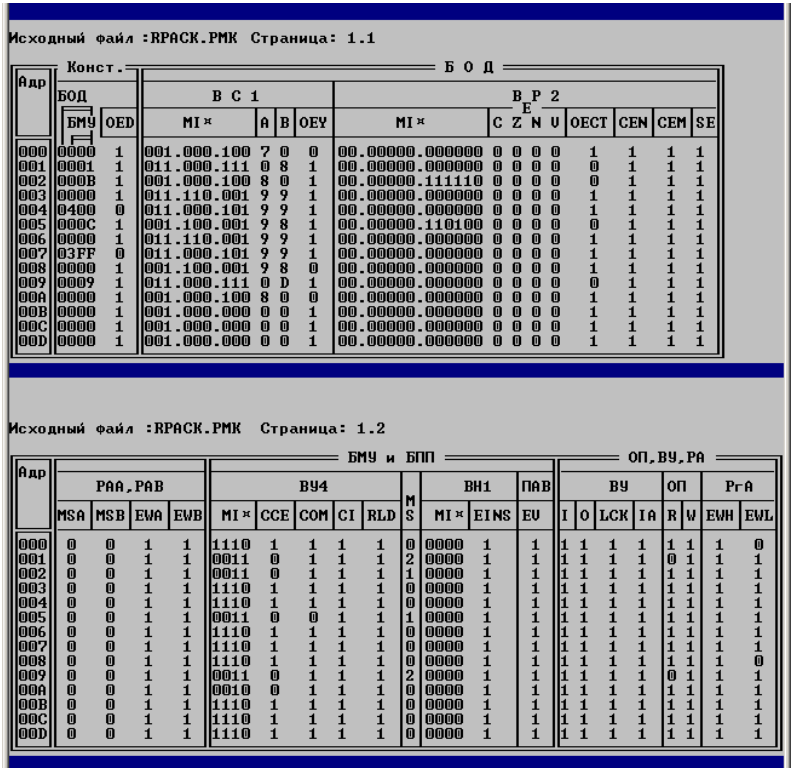


Рис. 2.47. ФС-мікроалгоритм переходу на МПП за кодом операції

```
      /-- Область налагодження зв'язків
      link l1: ct
      link l2:rdm
      link ewh:16
      link m:z,z,z,15,14,13,12,11,z,z,z,z
      /-- Видача адреси команди у РА
0000  {add nil,r7,z;oey;ewl;}
      /-- Завантаження команди у РК
0001  {r;cjp rdm,cp;add r8,bus_d,z;}
      /-- Визначення ФК
0002  {add nil,r8,z;cjp no,dfk;
      /-- Визначення ТА}
0003  {xor r9,r9,r9;}
0004  {add r9,r9,400h;}
0005  {and nil,r9,r8;cjp not zo,kad;}
      /-- Завантаження операнда
0006  {xor r9,r9,r9;}
0007  {add r9,r9,3ffh;}
0008  {and nil,r9,r8;oey;ewl;}
0009  {r;cjp rdm,cp;add r13,bus_d,z;}
      /-- Перехід за КОП
000A  {add nil,r8,z;oey;jmap;}
000B  {dfk}/--МПП розпакування двоадресної команди
000C  {kad}/--МПП завантаження для непрямої адресації
000D  {}
```

Рис. 2.48. Мікропрограма виконання одноадресної команди



Исходный файл :RPACK.PMK Страница: 1.1

Конст. — Б О Д

Адр	БОД БМУ	БОД OED	В С 1 MI×	В С 1 A	В С 1 B	В С 1 OEY	В Р 2 MI×	В Р 2 C	В Р 2 Z	В Р 2 N	В Р 2 V	В Р 2 OECT	В Р 2 CEN	В Р 2 CEM	В Р 2 SE
000	0000	1	001.000.100	7	0	0	00.00000.000000	0	0	0	0	1	1	1	1
001	0001	1	011.000.111	0	8	1	00.00000.000000	0	0	0	0	0	1	1	1
002	000B	1	001.000.100	8	0	1	00.00000.111110	0	0	0	0	0	1	1	1
003	0000	1	011.110.001	9	9	1	00.00000.000000	0	0	0	0	1	1	1	1
004	0400	0	011.000.101	9	9	1	00.00000.000000	0	0	0	0	1	1	1	1
005	000C	1	001.100.001	9	8	1	00.00000.110100	0	0	0	0	0	1	1	1
006	0000	1	011.110.001	9	9	1	00.00000.000000	0	0	0	0	1	1	1	1
007	03FF	0	011.000.101	9	9	1	00.00000.000000	0	0	0	0	1	1	1	1
008	0000	1	001.100.001	9	8	0	00.00000.000000	0	0	0	0	1	1	1	1
009	0009	1	011.000.111	0	D	1	00.00000.000000	0	0	0	0	0	1	1	1
00A	0000	1	001.000.100	8	0	0	00.00000.000000	0	0	0	0	1	1	1	1
00B	0000	1	001.000.000	0	0	1	00.00000.000000	0	0	0	0	1	1	1	1
00C	0000	1	001.000.000	0	0	1	00.00000.000000	0	0	0	0	1	1	1	1
00D	0000	1	001.000.000	0	0	1	00.00000.000000	0	0	0	0	1	1	1	1

Исходный файл :RPACK.PMK Страница: 1.2

БМУ и БПП — ОП,ВУ,РА

Адр	РАА,РАВ MSA	РАА,РАВ MSB	РАА,РАВ EWA	РАА,РАВ EWB	ВУ4 MI×	ВУ4 CCE	ВУ4 COM	ВУ4 CI	ВУ4 RLD	MS	ВН1 MI×	ВН1 EINS	ПАВ EV	ВУ I	ВУ O	ВУ LCK	ВУ IA	ОП R	ОП W	РгА EWH	РгА EWL
000	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	0
001	0	0	1	1	0011	0	1	1	1	2	0000	1	1	1	1	1	1	0	1	1	1
002	0	0	1	1	0011	0	1	1	1	1	0000	1	1	1	1	1	1	1	1	1	1
003	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
004	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
005	0	0	1	1	0011	0	0	1	1	1	0000	1	1	1	1	1	1	1	1	1	1
006	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
007	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
008	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	0
009	0	0	1	1	0011	0	1	1	1	2	0000	1	1	1	1	1	1	0	1	1	1
00A	0	0	1	1	0010	0	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
00B	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
00C	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
00D	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1

Рис. 2.49. Мікропрограма виконання одноадресної команди в кодах мікрокоманд

2.8. Розробка схем електричних принципових

Схеми електричні принципові блоку обробки даних та блоку мікропрограмного управління зображені на рис. 2.50 та рис. 2.51 відповідно. Принципи побудови та структурні схеми блоків розглянуті у розділах 2.4 та 2.5 відповідно. Загальна структурна схема МС зображена на рис. 2.41. Структурна схеми БОД зображена на рис. 2.7, шістнадцятирозрядний процесорний елемент у поєднанні зі схемою прискореного переносу – на рис. 2.21 та рис.2.23. Структурні схема БМУ наведена на рис. 2.27.

Приклад організації інтерфейсу модуля пам’яті, об’ємом $2^{m_2} \times 32$ (де 32 – розрядність даних) показаний на рис. 2.52.

За модульного принципу організації мікропроцесорної системи зовнішня загальна пам'ять системи являє собою набір однотипних модулів, підключених до системної шини. За цього, адреса, що містить m розрядів, розглядається як слово складене з двох полів: m_1 – старші розряди, що визначають номер модуля, та молодші m_2, що визначають адресу комірки в цьому модулі. Під час звернення до системної магістралі за певною адресою обирається тільки один модуль, номер якого співпадає із значенням в полі m_1. Номер модуля може задаватися, наприклад із допомогою мікротумблерів на платі. Обраний модуль приймає та передає значення у відповідності до виконуваного циклу, при цьому в якості адреси комірки пам'яті використовуються розряди m_2 адреси.

Модуль підключений до шини адреси (ША) та шини даних (ШД) через шинні формувачі (ШФ). Компаратор, що виконує функції селектора адреси (СА) формує сигнал вибору модуля (ВМ), якщо номер модуля співпадає зі значенням старших розрядів адреси m_1 Сигнал ВМ підключає модуль до ШД. В циклі читання дані із відповідної комірки видаються на шину даних, в циклі запису – записуються в комірку пам'яті.

У розглянутому інтерфейсі модуля пам'яті реалізована передача даних змінної довжини. Максимальна ширина вибірки даних – тридцять два розряди (чотири байти). Адреса слова у цьому випадку формується за формулою $A_{i+1} = A_i + LET$, де $LET = 4$ – кількість байтів у слові, $i = \overline{0,(2^{m_2} - 1)}$, m_2 – розрядність адреси. При цьому $LET = 2^n$ (n = 0, 1, 2, 3, 4, …) і значення n визначає кількість молодших розрядів адреси, що відповідають номеру байта у слові, а старша частина адреси відповідає адресі слова. За рахунок застосування чотирьох мікросхем пам'яті об'ємом $2^{m_2} \times 8$, за випадку $LET = 4$, можливий вибір даних як байтами, так і словами (по два байти) або подвійними словами (чотири байти). Блок управління (БУ) управляє комутатором, що складається з набору шинних формувачів, для підключення мікросхем пам'яті до ШД. На вхід блоку управління подаються молодші розряди (a_1, a_0) адреси m_2 ($LET = 2^2$, $n = 2$) та сигнали S_1, S_0, що визначають спосіб вибірки даних – байтами $(S_1S_0 = 01)$, словами $(S_1S_0 = 10)$ або подвійними словами $(S_1S_0 = 11)$. На виході БУ формуються сигнали включення мікросхемам пам'яті CS_3, CS_2, CS_1,CS_0, та сигнали включення шинних формувачів E_0, E_1, …, E_7.

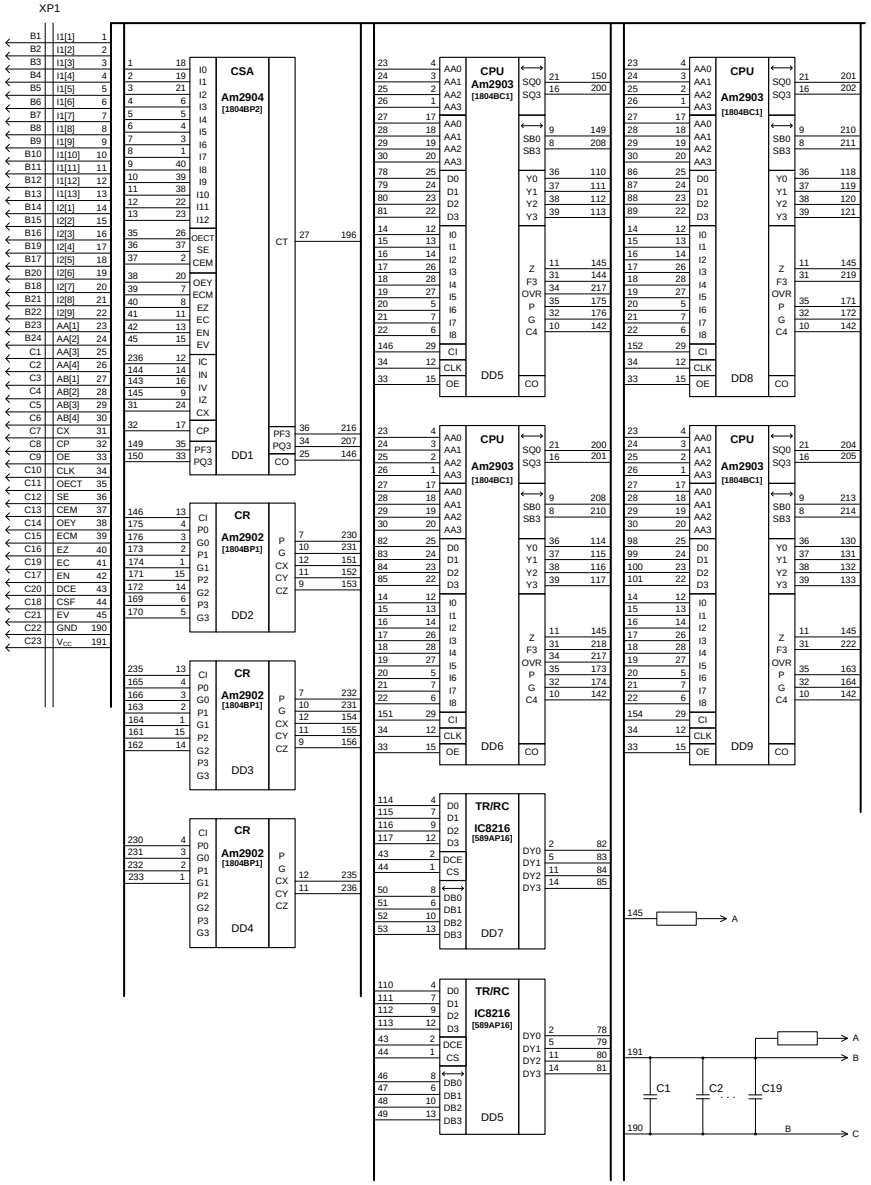
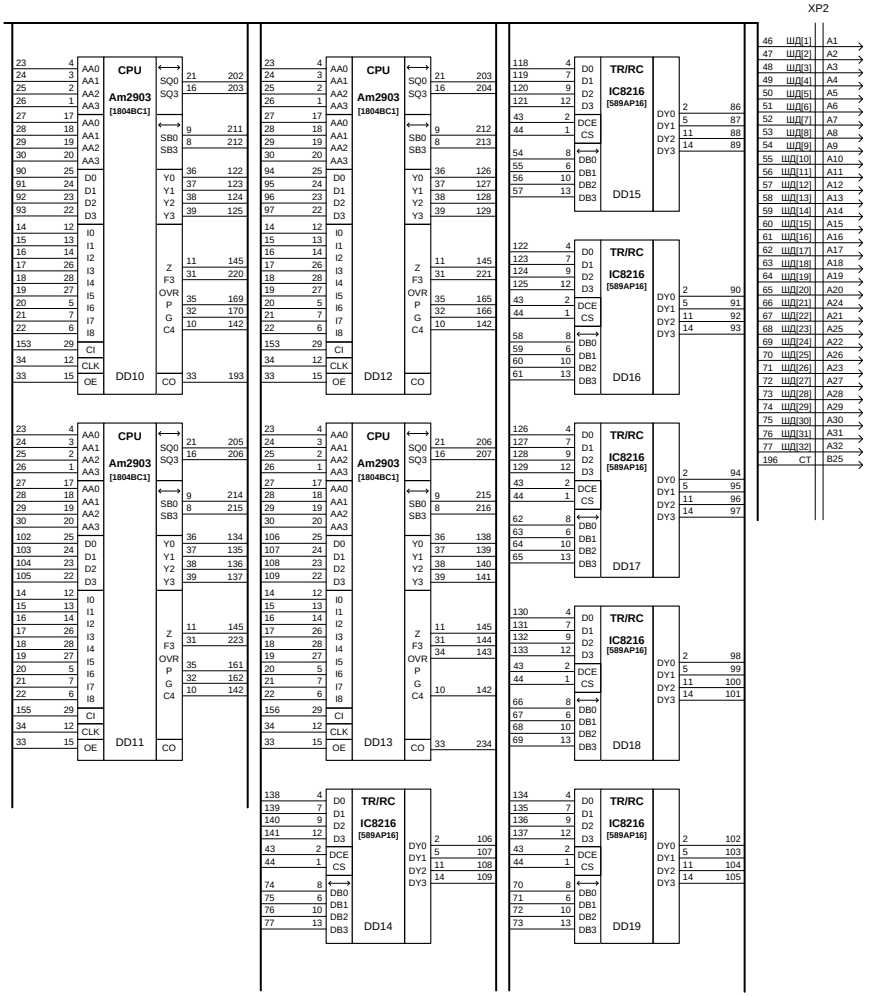


Рис. 2.50. Схема електрична принципова тридцятидвохрозрядного БОД



1. Виводи 10 мікросхем DD1, DD4-DD13, 16 мікросхем DD2-DD3, DD14-DD19 підключити до шини А(+5В).

2. Виводи мікросхем DD1, DD4-DD13, 8 мікросхем DD2- DD3, DD10-DD19 підключити до шини В (Загальн.).

Продовження рис. 2.50

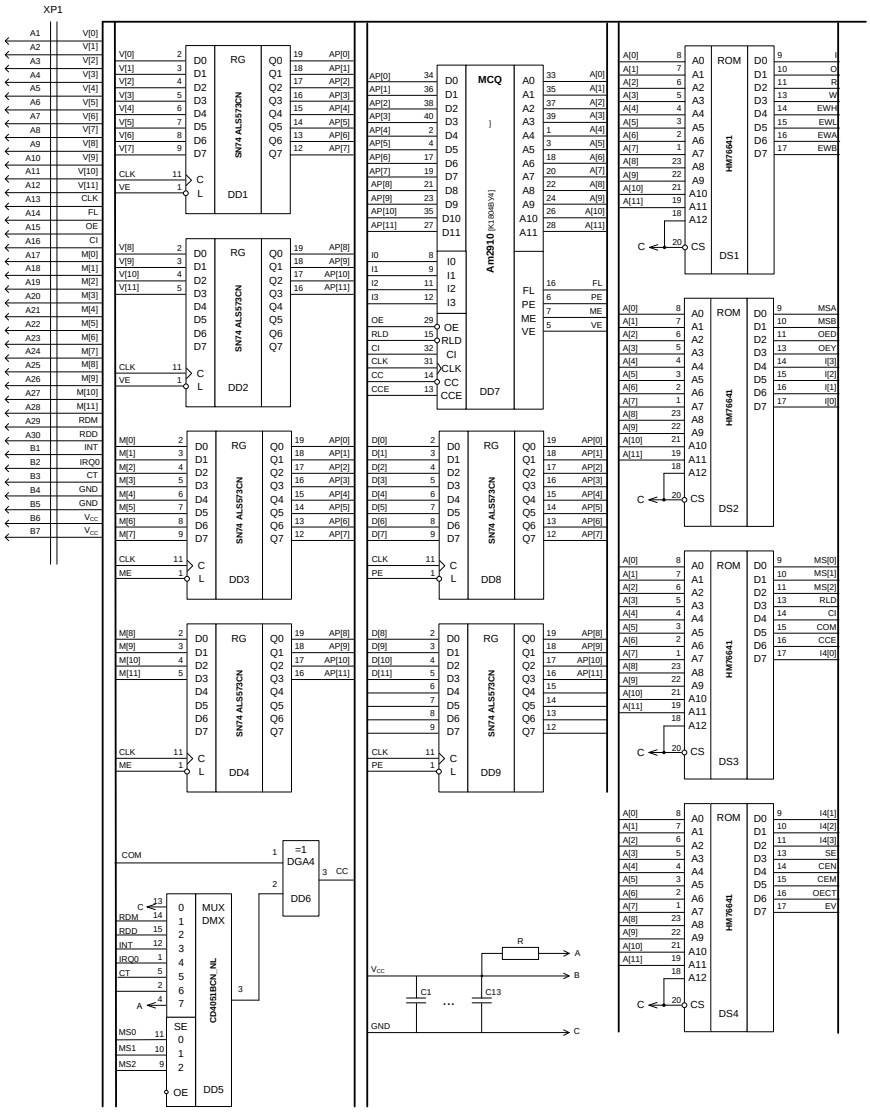
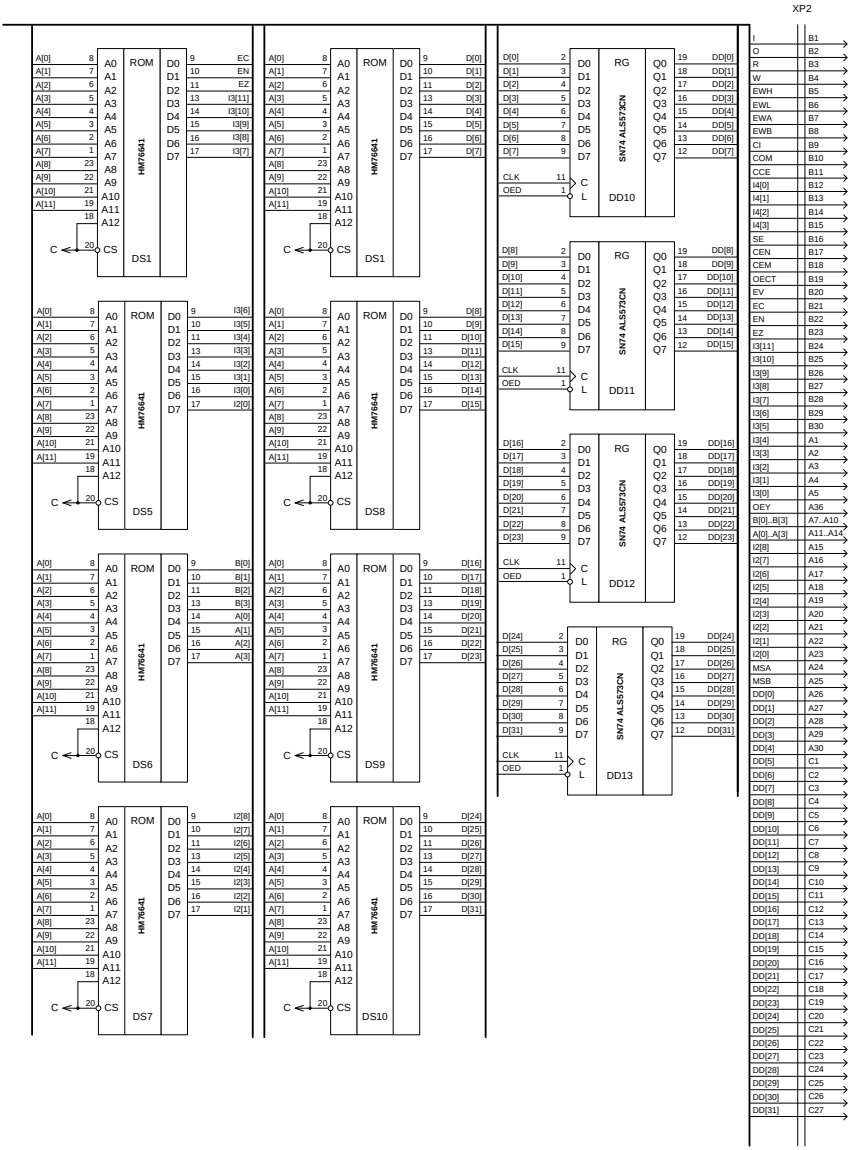


Рис. 2.51. Схема електрична принципова БМУ



1. Виводи 20 мікросхем DD1 – DD4, DD8 – DD9, DD10 – DD13, 30 мікросхеми DD7, 24 мікросхеми DS1 – DS12, 16 мікросхеми DD5 підключити до шини A (+5В).

2. Виводи 10 мікросхем DD1 – DD4, DD8 – DD9, DD10 – DD13, 10 мікросхеми DD7, 12 мікросхем DS1 – DS12, 8 мікросхеми DD5 підключити до шини B (Загальн.).

Продовження рис. 2.51

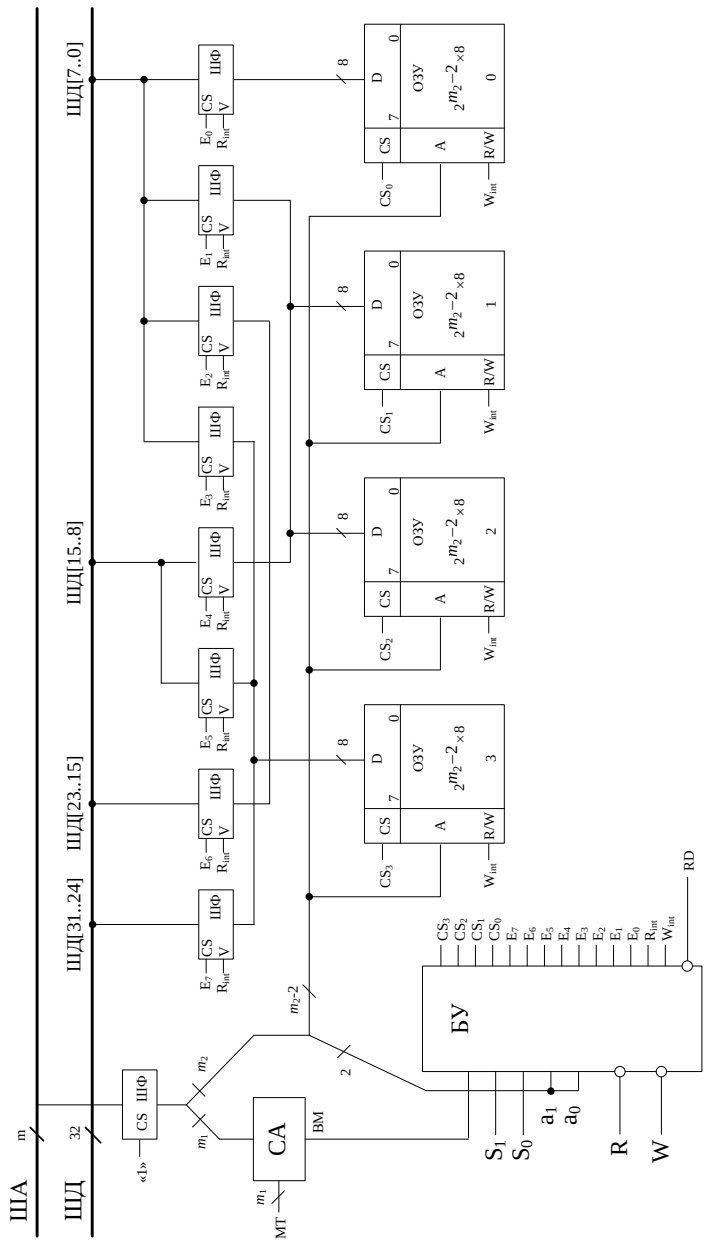


Рис. 2.52. Структурна схема модуля пам'яті із вибіркою слів змінної довжини

На вхід БУ також поступають сигнали читання *R*, запису *W*, вибору модуля ВМ. Логічні ланцюги формування сигналів R_{int}, W_{int} (внутрішні сигнали читання/запису), *RD* (сигнал готовності) умовно включені у склад блока управління. Детально структура інтерфейсу модуля пам'яті розглянуто у роботі [5].

Таблиця істинності блока управління наведена у табл. 2.35.

Таблиця 2.35. Таблиця істинності блоку управління

S_1	S_0	a_1	a_0	CS_3	CS_2	CS_1	CS_0	E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0
0	0	0	0	*	*	*	*	*	*	*	*	*	*	*	*
0	0	0	1	*	*	*	*	*	*	*	*	*	*	*	*
0	0	1	0	*	*	*	*	*	*	*	*	*	*	*	*
0	0	1	1	*	*	*	*	*	*	*	*	*	*	*	*
0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1
0	1	0	1	0	0	1	0	0	0	0	0	0	0	1	0
0	1	1	0	0	1	0	0	0	0	0	0	0	1	0	0
0	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	1
1	0	0	1	*	*	*	*	*	*	*	*	*	*	*	*
1	0	1	0	1	1	0	0	0	0	1	0	0	1	0	0
1	0	1	1	*	*	*	*	*	*	*	*	*	*	*	*
1	1	0	0	1	1	1	1	1	1	0	1	0	0	0	1
1	1	0	1	*	*	*	*	*	*	*	*	*	*	*	*
1	1	1	0	*	*	*	*	*	*	*	*	*	*	*	*
1	1	1	1	*	*	*	*	*	*	*	*	*	*	*	*

* – невизначений стан

За результатами синтезу отримані наступні логічні вирази, за якими побудовано функціональну схему блока управління (рис. 2.53).

$$CS_3 = S_1S_0 \vee a_1a_0 \vee \overline{S_0}a_1$$
$$CS_2 = S_0S_1 \vee a_1\overline{a_0}$$
$$CS_1 = S_1\overline{a_1} \vee \overline{a_1}a_0$$
$$CS_0 = E_0 = \overline{a_1}\,\overline{a_0}$$
$$E_7 = E_6 = S_1S_0$$
$$E_5 = S_1a_1$$
$$E_4 = S_1\overline{a_1}$$
$$E_3 = a_1a_0$$
$$E_2 = a_1\overline{a_0}$$
$$E_1 = \overline{a_1}a_0$$

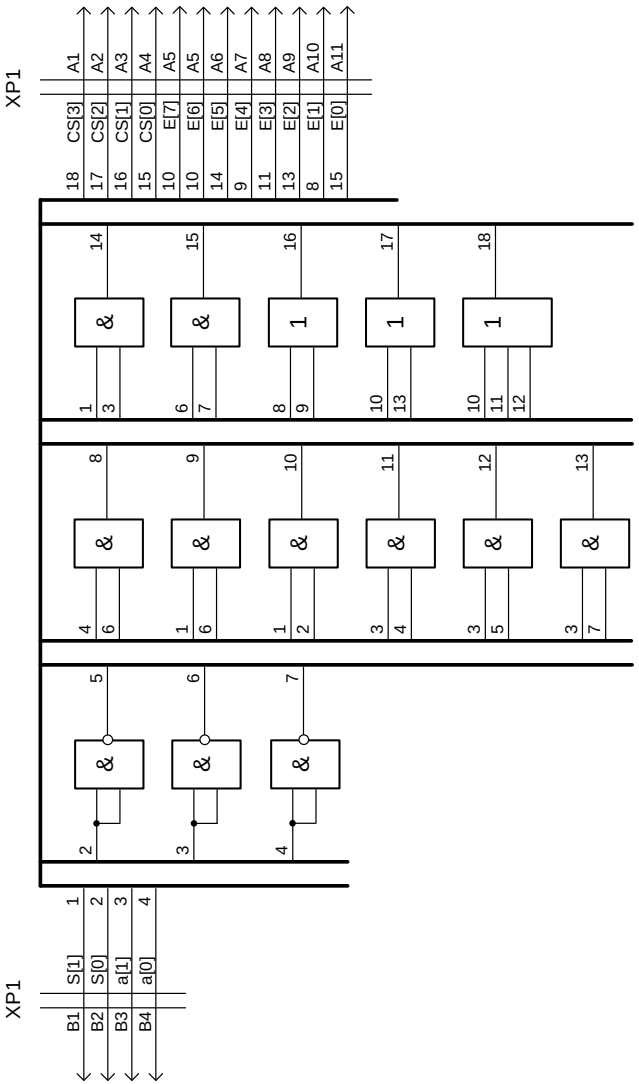


Рис. 2.53. Функціональна схема блока управління інтерфейсу модуля пам'яті

3. ОДНОКРИСТАЛЬНИЙ МІКРОКОНТРОЛЕР КР1816ВЕ48

3.1. Загальна характеристика

ІС серії 1816ВЕ призначені для побудови восьмирозрядних МПС, здатних виконувати фіксоване число програм, закладених в постійну пам'ять. Іншою назвою вказаних ІС є «мікроконтролер», яка визначає основну область застосування ІС – управління різноманітними технічними об'єктами.

Серія ІС КМ1816 включає декілька сімейств мікроконтролерів. В даному розділі в якості базової розглядається ІС КМ1816ВЕ48. Логічна організація, система команд і засоби введення-виведення інформації вказаної БІС аналогічні ІС ВЕ35, ВЕ39 та ВЕ49 цієї ж серії. Всі БІС сімейства пристосовані до ефективного рішення задач управління та регулювання. Відмінність їх складається в організації резидентної пам'яті програм (РПП), об'ємі внутрішньої пам'яті даних (РПД) і частоті тактових сигналів (табл.3.1.).

Таблиця 3.1. Параметри мікросхем

МК	Об'єм РПП, байт	Тип ЗПП	Об'єм РПД, байт	Тактуюча частота
ВЕ35	–	–	64	6 МГц
ВЕ48	1К	УФППЗУ	64	6 МГц
ВЕ39	–	–	128	11 МГц
ВЕ49	2К	ПЗУ	128	11 МГц

3.2. Архітектура мікроконтроллера КР1816ВЕ48

3.2.1. Умовне графічне позначення мікросхеми

Умовно графічне позначення мікросхеми МК48 наведено на рис. 3.1. Мікроконтролер конструктивно розміщений в корпусі з сорока виводами, сигнали на яких електрично-сумісні з елементами ТТЛ.

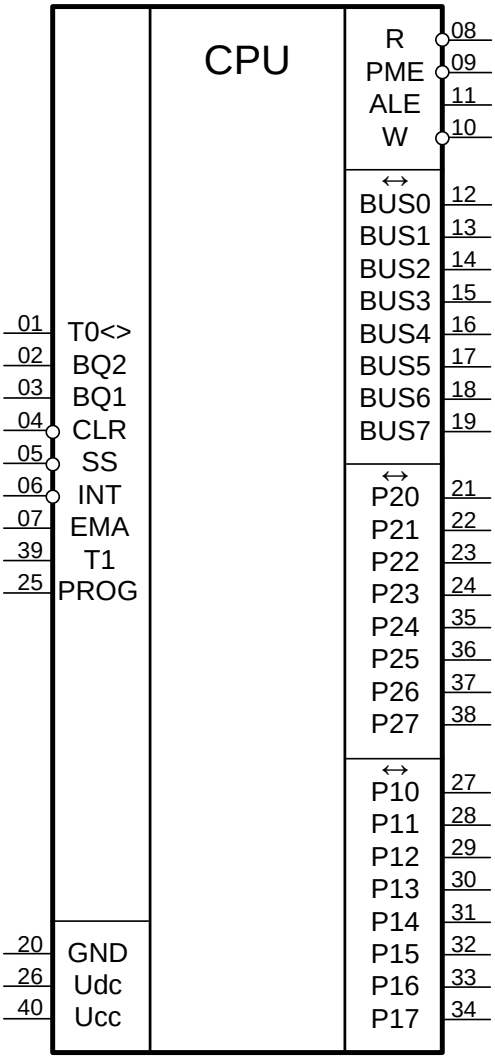


Рис. 3.1. Умовно графічне позначення мікроконтролера КР1816ВЕ48

Нижче приводяться символічні імена виходів мікросхеми.

*T*0 – вхід, що тестується командами *JT*0, *JNT*0; вихід тактового сигналу *CLK*;

*BQ*1 – вхід для підключення зовнішнього джерела синхронізації (для серії КР1830) або кварцового

		резонатора;
*BQ*2	–	вхід для підключення зовнішнього джерела синхронізації (для серії КР1830) або кварцового резонатора;
CLR	–	скидання;
SS	–	покрокове виконання програм;
INT	–	переривання;
EMA	–	встановлення режиму роботи з зовнішньою пам'яттю програм;
*T*1	–	вхід, що тестується командами *JT*1, *JNT*1; вхід зовнішніх подій;
PROG	–	вихід стробу для розширювача вводу/виводу;
R	–	читання зовнішньої пам'яті даних;
PME	–	дозвіл читання зовнішньої пам'яті програм;
ALE	–	строб адреси зовнішньої пам'яті;
W	–	запис у зовнішню пам'ять даних;
GND	–	загальний;
Vcc	–	напруга живлення +5В;
UDC	–	додаткова напруга живлення +5В;

*BUS*0 – *BUS*7	–	шина даних; порт *BUS*;
*P*10 – *P*17	–	восьмирозрядний порт вводу-виводу;
*P*20 – *P*27	–	восьмирозрядний порт вводу-виводу;

3.2.2. Структура мікроконтролера

Структурна схема МК48 наведена на рис. 3.2. Мікроконтролер містить резидентну пам'ять програм (РПП), резидентну пам'ять даних (РПД); пристрій управління і синхронізації, до складу якого входить лічильник команд, регістр команд і регістри ознак; арифметико-логічний пристрій, до складу якого входить АЛБ, акумулятор і регістри; регістр слова стану програми; таймер/лічильник. Обмін даними здійснюється через порти *P*1, *P*2 і *BUS*.

Модель програміста МК48 зображена на рис. 3.3. На моделі програміста застосовуються наступні умовні позначення:

PC	–	лічильник команд;
T		допоміжний регістр;

A	– акумулятор,
TCNT	– таймер/лічильник;
RSW	– регістр слова стану програми;
MB	– ознака банку пам’яті;
TF	– ознака переповнювання таймера;
*F*1	– ознака користувача;
C	– ознака переносу;
AC	– ознака додаткового переносу;
*F*0	– ознака користувача;
RB	– ознака банку регістрів;
SP	– покажчик стеку;
*P*1	– порт вводу/виводу
*P*2	– порт вводу/виводу
BUS	– порт вводу/виводу

Акумулятор

Акумулятор *A* являється восьмирозрядним регістром, який крім вказівника адреси, використовується в якості приймача або джерела операнда.

Арифметико-логічний пристрій

До складу АЛП входять наступні блоки: комбінаційна схема обробки байтів, регістри *T*, регістр-акумулятор *A*, схема десяткового коректора і схема формування ознак .

Акумулятор використовується як регістр операнда і регістра результату. Регістр тимчасового зберігання операнда *T*1 програмно недоступний і використовується для тимчасового зберігання другого операнда при виконанні двоопераднних команд. Комбінаційна схема АЛП може виконувати наступні операції: складання байтів з перенесенням або без нього; логічні операції І, АБО та ВИКЛЮЧНЕ АБО; інкремент, декремент, інверсія, циклічний зсув вліво і вправо із встановленням (або без встановлення) ознаки переносу, обмін тетрадами в байті; десяткова корекція вмісту акумулятора.

Під час виконання команди `JBb` (де $b = \overline{0,7}$) в АЛБ формується ознака *Bb*, яка відповідає значенню відповідного розряду акумулятора (рис. 3.3). Під час виконання команд `JZ`, `JNZ`, `DJNZ` формується ознака *z* нульового вмісту акумулятора або зазначеного

у команді регістру. За значеннями ознак *Bb* і *z* здійснюється розгалуження програм. Тригери для зберігання ознак *Bb* і *z* відсутні тобто після виконання команд ознаки не запам’ятовуються.

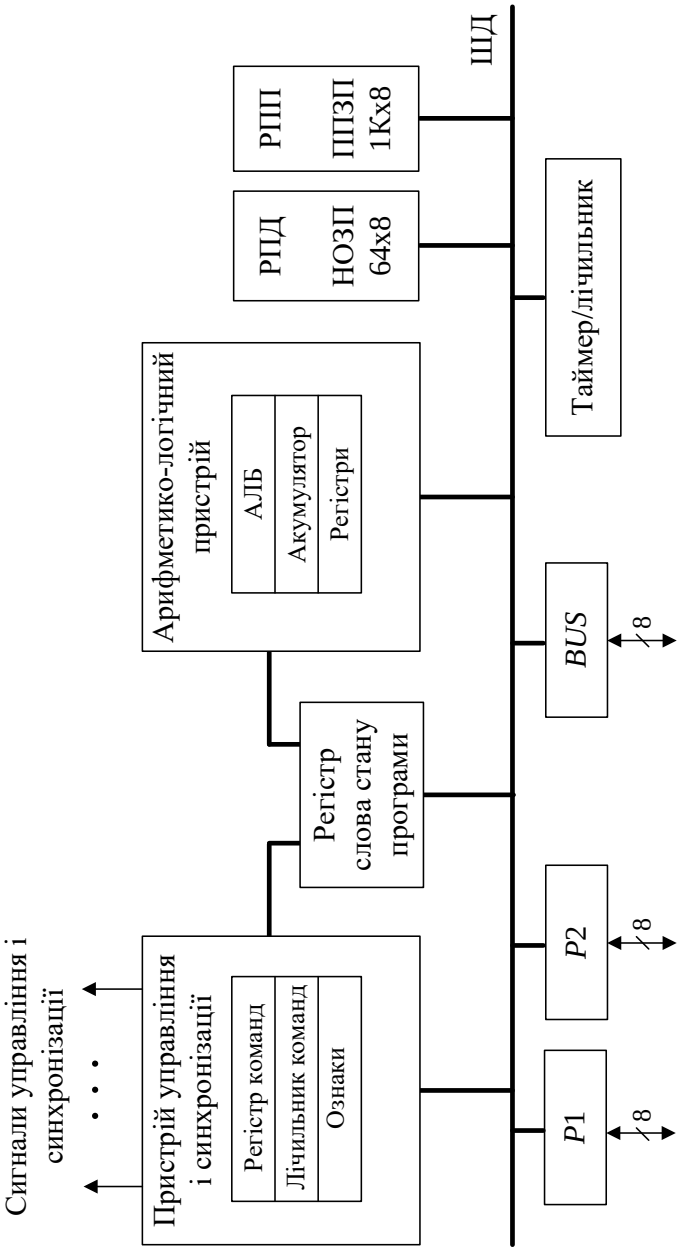


Рис. 3.2. Структурна схема мікроконтролера КР1816ВЕ48

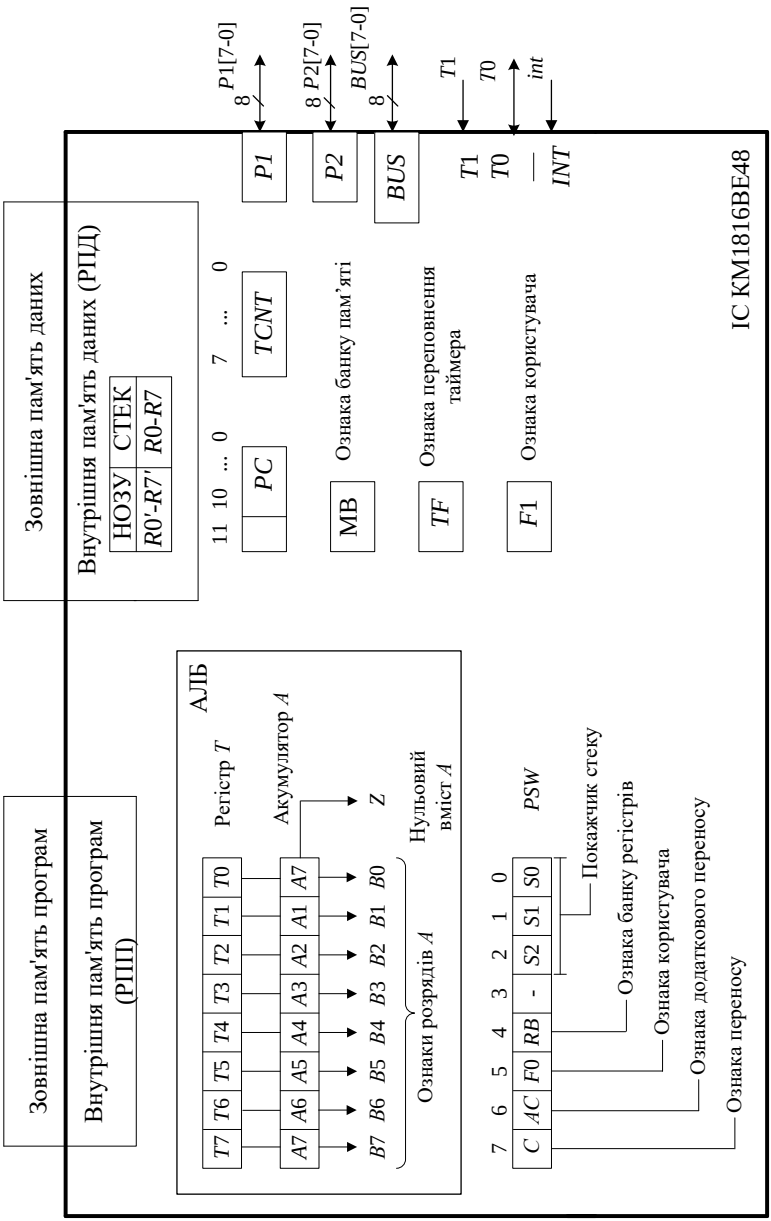


Рис. 3.3. Модель програміста

При виконанні операцій обробки даних в АЛБ виробляються ознаки, які формуються на комбінаційній схемі і не фіксуються на тригерах, за винятком ознаки переносу *C*.

Ознака додаткового переносу *AC* встановлюється під час переносу з молодшої тетради в старшу. Ознаки переносу і допоміжного переносу фіксуються на тригерах, що входять до складу регістра слова стану програми *PSW*. Формат регістра *PSW* показаний на рис. 3.3. Окрім перерахованих ознак логіка умовних переходів МК оперує прапорами *F*0 і *F*1, функціональне призначення яких визначається розробником; прапором переповнювання таймера *TF*, сигналами на входах *T*0 і *T*1. Програмістом можуть бути також використані ознаки робочого банку регістрів *RB* і вибраного банку зовнішньої пам'яті програм МВ. Крім того, логікою переходів після закінчення кожного машинного циклу обпитується ще одна ознака, а саме ознака дозволу/заборони переривання.

Лічильник команд

Лічильник команд *PC* має довжину дванадцять розрядів. Після вибірки чергової команди вміст *PC* збільшується на одиницю. Перенос при цьому розповсюджується тільки від нульового до десятого розряду.

Старший розряд *PC*[11] виконує спеціальну функцію і визначається ознакою банку пам’яті програм *MB*, яка встановлюється програмно, за допомогою команд `SFL MBO` і `SEL MB1`. Програмні засоби для перевірки ознаки банка пам’яті *MB* відсутні.

Пам’ять програм

Пам'ять програм і пам'ять даних в МК48 розділені.

Пам'ять програм розділяється на резидентну, розташовану всередині ІС, та зовнішню, для реалізації якої необхідні додаткові ІС пам’яті. Максимальний адресний простір пам’яті програм складає 4 Кб (рис. 3.4). Резидентна пам'ять програм (РПП) представляє собою перепрограмоване ПЗУ(ППЗУ) об’ємом 1 Кб (адреса від 0 до 1023).

Адреси 0, 0003 та 0007 мають спеціальне призначення. З адреси 0 починається виконання програми за системним скиданням. Комірка 0003 призначена для зберігання початкової адреси підпрограми обслуговування зовнішнього переривання, а комірка

0007 – зберігання початкової адреси підпрограми обробки переривання від таймера/лічильника.

Пам'ять програм розглядається як два банки – нульовий банк програм і перший банк програм. Якщо встановлюється розряд лічильника команд *PC*[11] = 0, то вибір слів здійснюється із нульового банку пам'яті (адреси від 0 до 2047), і, якщо *PC*[11] = 1 – із першого банку пам'яті (адреси від 2047 до 4095). Вибір банку пам'яті здійснюється командами `SEL MBO` та `SEL MB1`, які встановлюють ознаку *MB* вибору банку пам'яті.

Окрім розділення на банки програм, пам'ять програм поділяється на сторінки об'ємом 256 байт. Це пов'язано з тим, що команди умовних переходів модифікують тільки вісім молодших розрядів адреси, тобто забезпечують перехід всередині сторінки. При переході до підпрограм обслуговування переривань автоматично встановлюється в нуль розряд *PC*[11] лічильника команд. У зв'язку з цим підпрограми обслуговування переривань повинні розміщуватися в нульовому банку пам'яті.

Способи адресації операндів в пам'яті програм:

– безпосередня;
– непряма з використанням акумулятора.

В першому випадку операнд розміщується в байті, наступному за кодом команди. За непрямої адресації, в якості покажчика адреси операнда застосовується акумулятор *A*. Акумулятор містить адресу чергової сторінки пам'яті (`MOVP A,@A`), або третьої сторінці (`MOV3P A,@A`).

Пам'ять даних

Пам'ять даних розділяється на внутрішню і зовнішню. Внутрішня пам'ять даних представляє собою ОЗП ємністю 64 байти (рис. 3.5). Пам'ять даних містить два банки регістрів загального призначення. Банк регістрів 0 включає регістри $R0 - R7$ з адресами 0 – 0007, а банк регістрів 1 – регістри $R0 - R7$, які мають адреси 0024 – 0031. Вибір банку регістрів здійснюється командами `SEL RBO` та `SEL RB1`, які встановлюють ознаку *RB*, що знаходиться в четвертому розряді *PSW*.

Спеціальна команда для перевірки *RB* відсутня, але ознаку можна проаналізувати, перенісши вміст *PSW* в *A* і виконати перехід за ознакою *B*4, яка перевіряється командою `JB4`. Комірки з адресами 0008 – 0023 можуть використовуватися як восьмирівневий

стек шістнадцятирозрядних слів або як комірки ОЗП даних з довільним доступом.

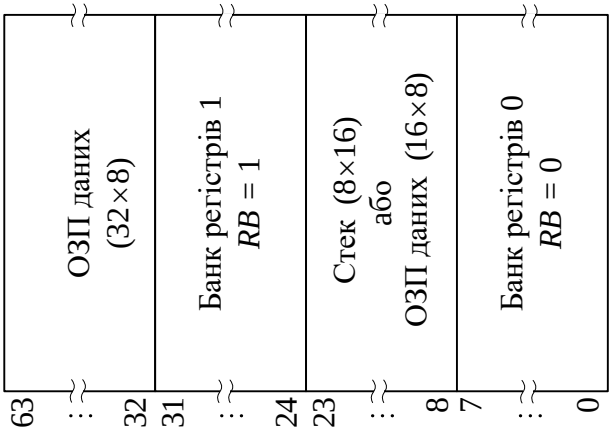


Рис. 3.5. Карта розподілу адрес внутрішньої пам'яті даних

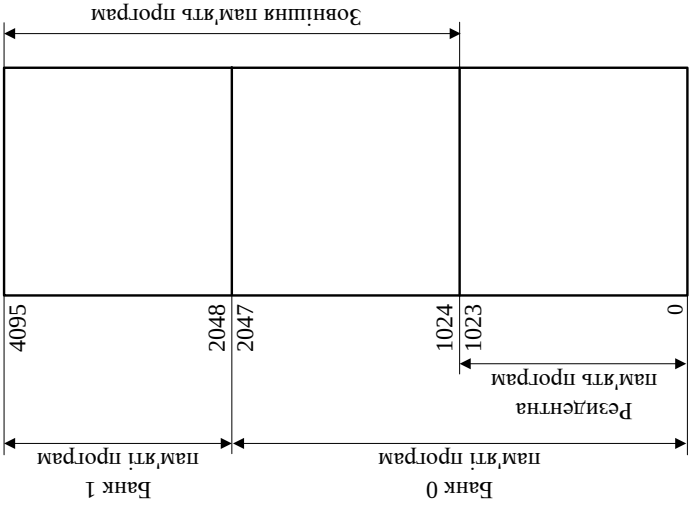


Рис. 3.4. Організація пам'яті програм

Покажчик стеку SP входить до складу регістра *PSW* (розряди *S*2 – *S*0). При переході до підпрограми в стек записується вміст *PC* і чотирьох старших розрядів *PSW* (рис. 3.6). Частина внутрішньої пам'яті з адресами 0032 – 0063 використовується тільки в якості ОЗП даних.

Способи адресації для доступу до внутрішньої пам'яті даних:

– пряма регістрова;

– непряма регістрова.

Пряма регістрова адресація використовується для звернення до регістрів загального призначення $R0 - R7$ банку регістрів 0, якщо $RB = 0$, та $R0 - R7$ банку регістрів 1, якщо $RB = 1$. В цьому випадку трирозрядна адреса регістра присутня в коді команди.

За допомогою непрямої регістрової адресації можна звернутися до будь-якого байту внутрішньої пам'яті даних (в тому числі до регістрових банків і стеку). В якості покажчика адресу операції в такому випадку використовуються регістри $R0$ і $R1$, відповідно до обраного банку регістрів (нульового або першого). Для обміну даними застосовуються наступні команди `MOV A,@Rr`, `MOV @Rr,A`, (де $r = 1, 0$).

Рис. 3.6. Організація стеку

Непряма регістрова адресація застосовується також під час виконання команд звернення до зовнішньої пам'яті даних. Виходячи з того, що в якості покажчика адреси використовуються восьмирозрядні регістри $R0$, $R1$ обраного банку регістрів, максимальний об'єм зовнішньої пам'яті даних складає 256 байт. Під

час обміну з зовнішньою пам'яттю даних застосовуються команди `MOVX A,@Rr`, `MOVX @Rr,A`, (де $r = 1, 0$).

Стек

Внутрішній восьмирівневий стек забезпечує автоматичне збереження і відновлення основних параметрів обчислювального процесу при запитах переривання і при поверненні після обслуговування переривання.

Система переривань

В МК48 реалізована *система переривань* від двох джерел: внутрішнього таймера/лічильника і зовнішніх джерел переривань, наприклад від контролеру пріоритетних переривань.

Таймер/лічильник

Таймер/лічильник TCNT являється восьмирозрядним лічильником, що підсумовує, який можна читати і завантажувати через акумулятор *A*, використовуючи відповідні команди `MOV`. Він може працювати в режимі таймера і в режимі лічильника. В режимі таймера на вхід *TCNT* через дільники частоти поступають сигнали з частотою *F*/480, де *F* – частота, що задається кварцовим резонатором або зовнішнім генератором. Наприклад, при *F* = 6 МГц лічильник збільшує свій стан на 1 через кожні 80 мкс. Шляхом встановлення лічильника у певний вихідний стан і аналіз його переповнення можуть бути реалізовані різні часові затримки. Якщо 256 станів не забезпечують бажану затримку, то можна розрахувати декілька періодів роботи *TCNT*, накопичуючи в робочому регістрі необхідне число переповнень лічильника. Під час переходу *TCNT* із стану 255 в стан 0 ознака *TF* встановлюється в одиницю. Ця ознака може бути проаналізована командою `JTF`. Крім того, якщо переривання від *TCNT* дозволено командою `EN TCNT`, то при встановленні *TF* в 1 здійснюється перехід до підпрограми обслуговування переривання за вектором 0007. Переривання від *TCNT* може бути не дозволено командою `DIS TCNTI`. Після виконання команди `JTF` і при переході до підпрограми обслуговування переривання *TF* переходить в 0.

В режимі лічильника подій *TCNT* збільшує свій стан на 1 кожен раз, коли сигнал на вході *T*1 переходить із стану 1 в стан 0. В режимі таймера *TCNT* запускається командою `STRT T`, а в режимі

лічильника – командою `STRT CNT`. Зупинка *TCNT* здійснюється командою `STOP TCNT` або системним скиданням.

Регістр слова стану програми

Крім описаних ознак *MB, RB*, і *TF* є також внутрішні ознаки *C, AC* і *F*0, які зберігаються в *PSW*, і ознака *F*1, яка до складу *PSW* не входить (не запам'ятовується у стеку при переході до підпрограм). Ознака *C* встановлюється у відповідності з переносом із сьомого розряду, яка формується при виконанні команд суми слів в арифметично-логічному пристрої. Вона встановлюється також при виконанні команд зсуву вмісту *A* в поєднані з бітом *C*. Ознака *AC* встановлюється при виконанні команд суми, в залежності від значення переносу із третього розряду (переносу між тетрадами). Значення *AC* використовується командою `DA`, яка призначена для корекції вмісту акумулятора під час підсумовування двійково-десяткових чисел. Ознаки *F*0 і *F*1 являються ознаками користувача. Існують і зовнішні ознаки *T*0, *T*1 і *INT*, які формуються поза ІС.

Під час виконання команд умовного переходу перевіряються ознаки *Z, Bb, C, F*0, *F*1, *T*0, *T*1, *TF* і *INT*. Виводи *T*0, *T*1 і *INT* можуть використовуватися з іншою метою: *T*0 – в якості видачі на об'єкт управління зовнішніх сигналів синхронізації з частотою *F*/3 (за командою `ENTO CLK`), *T*1 – в якості входу таймера/лічильника (`STRT CNT`), *INT* – в якості входу зовнішнього переривання (`ENI`). Ряд ознак, а саме: *C, F*0, *F*1, *RB* і *MB* можна встановлювати програмно.

Порти вводу/виводу

Мікроконтролер вміщує три *порти вводу/виводу*: *P*1, *P*2 і *BUS*. Порти *P*1 і *P*2 називають «квазідвоспрямованими». Їх особливість полягає в тому, що при вводі даних, над ними та поточним станом порту (даними, які виводилися із порту останніми) виконується порозрядна логічна операція І. Вихідні дані в порту запам'ятовуються. При скиданні системи кожному розряду порту присвоюється значення 1. У системі команд МК48 є команди, які дозволяють виконувати запис нулів і одиниць в будь-якому розряді або групі розрядів порту, але оскільки в цих командах маска задається безпосереднім операндом, то необхідно знати розподіл ліній, що скидаються і встановлюваних, на етапі розробки прикладної програми. В тому випадку, якщо маска обчислюється програмою і наперед не відома, в ОЗП необхідно мати копію стану

порту виведення. Ця копія по командах логічних операцій об'єднується з обчислюваною маскою в акумуляторі і потім завантажується в порт. Необхідність цієї процедури викликана тим, що в МК48 відсутня можливість виконати операцію читання значень портів *P*1 і *P*2 для визначення колишнього стану порту виведення. Порт *P*2 відрізняється від порту *P*1 тим, що його молодші чотири біта можуть бути використані для підключення додаткових портів *P*4, *P*5, *P*6, *P*7. Робота цих зовнішніх портів синхронізується сигналом `PROG`.

Порт *BUS* має звичайні двоспрямовані виходи з трьома станами. Порт застосовується для побайтного вводу/виводу даних. За допомогою команд `ORL` і `ANL` можливо маскувати байти, що передаються через порт, з ціллю обробляти у байті окремі біти або групу бітів. Команди звернення до портів включають безпосередній номер порту. У мікропроцесорних системах простої конфігурації, коли порт *BUS* не використовується як порт-розширювач системи, обмін виконується по командах `INS`, `OUTU` і `MOVX`. Можливе поперемінне використання команд `OUTL` і `MOVX`. Проте при цьому необхідно пам'ятати, що байт, який виводиться по команді `OUTL` фіксується в буферному регістрі порту *BUS*, а команда `MOVX` знищує вміст буферного регістра порту *BUS*. Команда `INS` не знищує вміст буферного регістра порту. В МПС що мають зовнішню пам'ять програм, порт *BUS* використовується для видачі адреси зовнішній пам'яті і для прийому команди із зовнішньої пам'яті програм. Отже, в таких системах використання команди `OUTL` позбавлене сенсу.

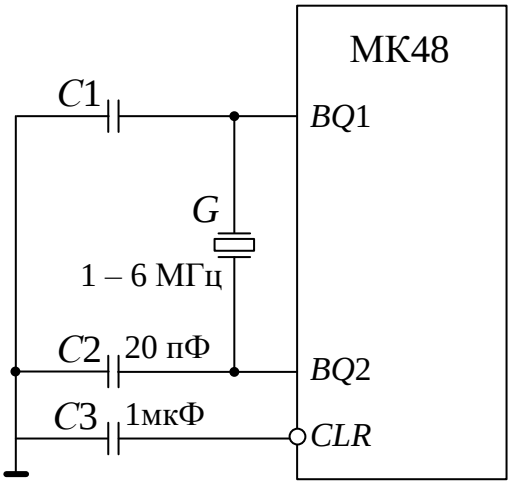
3.3. Основні режими роботи мікроконтролера КР1816ВЕ48

3.3.1 Ініціалізація системи

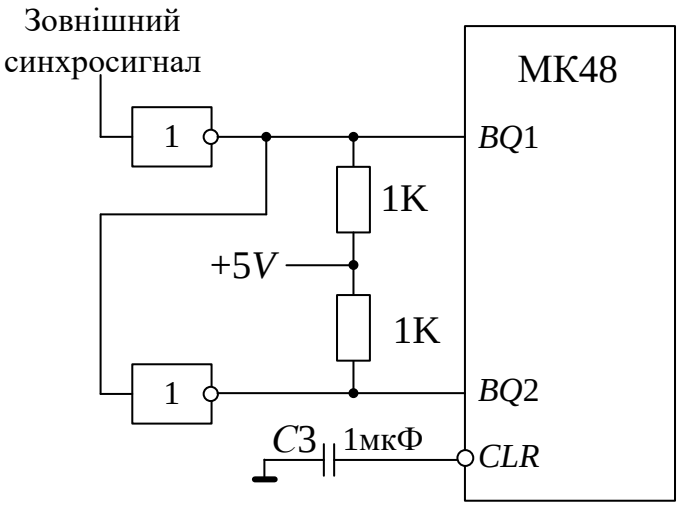
Частота синхронізації МК48 може бути задана за допомогою кварцового резонатора, схема включення якого вказана на рис. 3.7 *а*. Для синхронізації роботи МК48 з іншим обладнанням може також бути використаний зовнішній генератор (рис. 3.7. *б*).

Ініціалізація системи здійснюється за допомогою подачі на вхід *CLR* нульового сигналу, тривалістю не менш 50 мс. Ініціалізація системи може здійснюватись безпосередньо після включення

джерела живлення +5В при використанні схеми скидання (рис. 3.7 *а*), яка в найпростішому випадку складається із конденсатора *С*3.



а



б

Рис. 3.7. Схеми синхронізації і скидання мікроконтролера: а – підключення кварцового резонатора; б – підключення зовнішнього генератора

В процесі ініціалізації забороняються переривання, порти *P*1 та *P*2 налаштовуються на ввід інформації, виходи порту *BUS* переводяться в високоомні стани (за встановлення *EMA* = 0), зупиняється таймер/лічильник *TCNT*, забороняється видача тактуючих сигналів на виході *T*0. Встановлюються в нуль тригери *MB, TF, F*1, *F*0, *RB*, а також регістри *SP* та *PC*. В результаті цього спочатку вибираються нульові банки пам'яті програм і регістрів, покажчик стеку *SP* вказує на нульову комірку стеку (байти ЗПД з адресами 0008 і 0009) та виконання програми починається з нульової адреси.

3.3.2. Режими роботи з пам'яттю

Режим роботи з резидентною пам'яттю програм

У режимі роботи МК48 з резидентною пам'яттю програм (для ВІС ВЕ48 і ВІС ВЕ49 за *EMA* = 0) для зв'язку з об'єктом управління використовуються три порти (*P*1, *P*2, *BUS*). При цьому кожний розряд будь-якого порту може бути запрограмований на ввід чи вивід інформації. За необхідності між виходами МК48 і входами/виходами об'єкту управління включаються відповідні елементи, що забезпечують формування сигналів за різними параметрами. Під час звернення до резидентної пам'яті програм зовнішні управляючи сигнали, окрім сигналу *ALE*, не формуються.

Сигнал *ALE* за необхідності можна використовувати для синхронізації роботи внутрішніх пристроїв. Якщо адреса перевищує допустиме для резидентної пам'яті програм значення – 1023, то МК48 автоматично переходить в режим роботи з зовнішньою пам'яттю програм.

Режим роботи з зовнішньою пам'яттю програм

Режим роботи МК48 з зовнішньою пам'яттю програм можливий за застосування додаткових мікросхем ПЗП. Якщо не використовувати сторінкову адресацію, об'єм пам'яті програм можна розширити до 4К байт. За встановлення сигналу *EMA* = 1 доступні всі 4К байта зовнішньої пам'яті. Якщо сигнал *EMA* = 0, то адресація комірок зовнішньої пам'яті розпочинаються з адреси 1024. При цьому область пам'яті з адресами від 0 до 1023 належать резидентній пам'яті програм. Схема підключення зовнішньої пам'яті програм об'ємом 4К наведена на рис. 3.8, а часова діаграма читання байта (команди або даних) – на рис. 3.9.

Для підключення зовнішньої пам'яті програм використовуються виходи портів *BUS*[7..0] та *P*2[3..0]. Для зберігання адреси звернення до пам'яті використовується зовнішній регістр адреси РА.

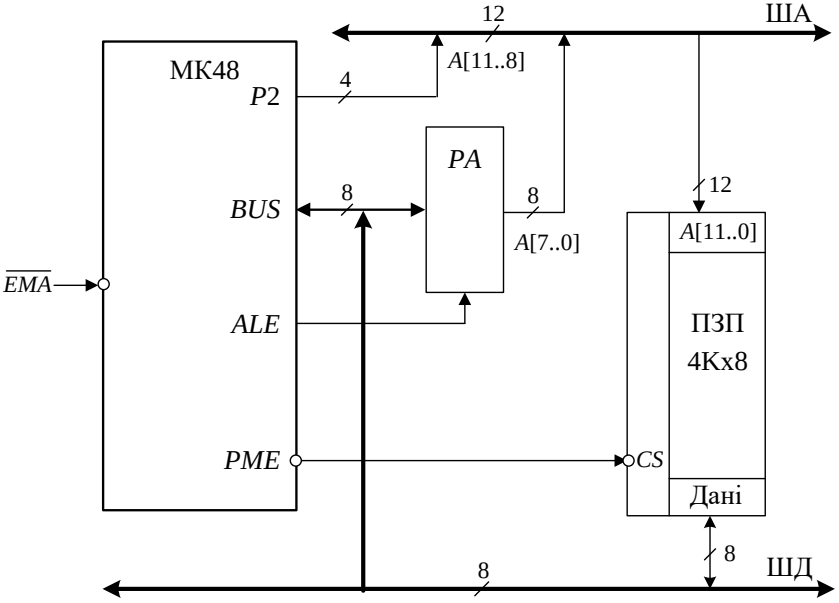


Рис. 3.8. Схема підключення зовнішньої пам'яті програм

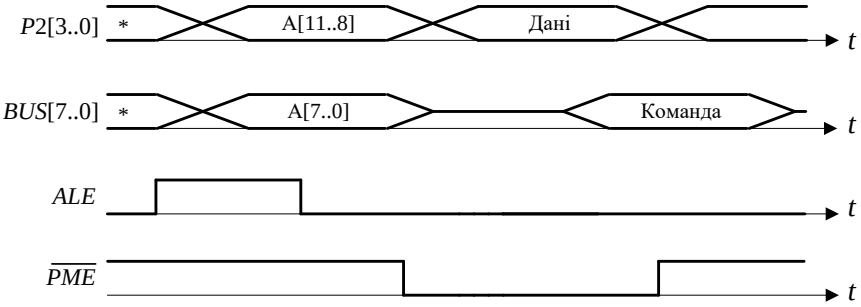


Рис. 3.9. Часова діаграма вибору інформації з зовнішньої пам'яті програм

В процесі звернення до зовнішньої пам'яті програм МК48 формує дванадцятирозрядну адресу, де старші розряди адреси *A*[11..8] формуються на виходах *P*2[3..0] мікросхеми, а молодші розряди *A*[7..0] – на виходах *BUS*[7..0]. Запис адреси в регістр адреси РА стробується сигналом *ALE*. Після цього виходи порту *BUS* перемикаються на ввід інформації та видається сигнал дозволу читання пам'яті *PME*. За цим сигналом на виходах даних мікросхеми пам'яті формується байт інформації (команди або даних), який за шиною *BUS* приймається в МК48.

За випадку, коли резидентна пам'ять програм відсутня (*EMA* = 0), необхідно передбачити логіку відключення резидентної пам'яті під час звернення до адрес 0 – 1023. Наприклад, якщо вибір зовнішньої пам'яті здійснюється низьким рівнем сигналу на вході *CS*, то на цей вхід можна подати значення логічної функції $NOT(P2[3]\,OR\,P2[2])$.

Режим роботи з зовнішньою пам'яттю даних

В режимі роботи МК48 з зовнішньою пам'яттю даних використовуються додаткові мікросхеми ОЗП об'ємом 256 байт.

Якщо адресний простір має об'єм більш ніж 256 байт, то необхідна сторінкова організація зовнішньої пам'яті даних. При роботі з адресами в межах однієї сторінки застосовуються команди `MOVX A,@Rr`, `MOVX @Rr,A` (де $r = 1, 0$). Обмін інформацією здійснюється між акумулятором А і коміркою ОЗП, яка непрямо адресується через регістр *R*0 або *R*1. Переключення між сторінками потребує використання додаткових команд вибору сторінки пам'яті даних.

На рис. 3.10 показаний спосіб підключення до МК48 *n* сторінок зовнішньої пам'яті даних з використанням *k* виходів порту *P*1 (де $k =]\log_2 n[\,)$ і дешифратора *DC*.

За наявності одночасно зовнішньої пам'яті програм і даних використовується один і той самий зовнішній регістр адреси.

Часові діаграми циклів звернення до зовнішньої пам'яті даних наведені на рис. 3.11.

Команди виконуються за два цикли. В процесі читання пам'яті спочатку на шину *BUS* встановлюється адреса, яка фіксується у зовнішньому регістрі РА за встановлення сигналу *ALE*. Після чого виходи порту *BUS* перемикаються для вводу інформації і формується сигнал читання даних *R*. За цим сигналом ЗПД

виставляє дані на ШД, які через порт *BUS* приймаються в МК48. На протязі циклу запису інформація, що передається, встановлюється на шині *BUS*. Запис в пам'ять стробується сигналом *W*.

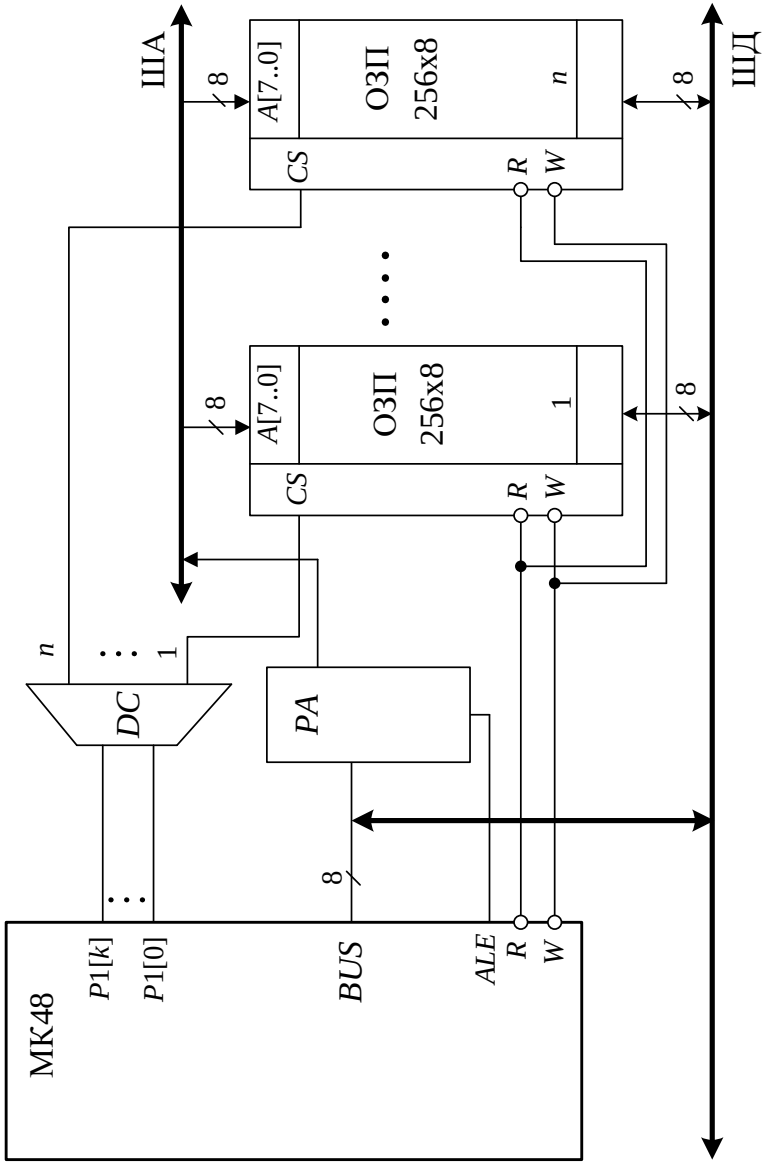


Рис. 3.10. Схема підключення зовнішньої пам’яті даних

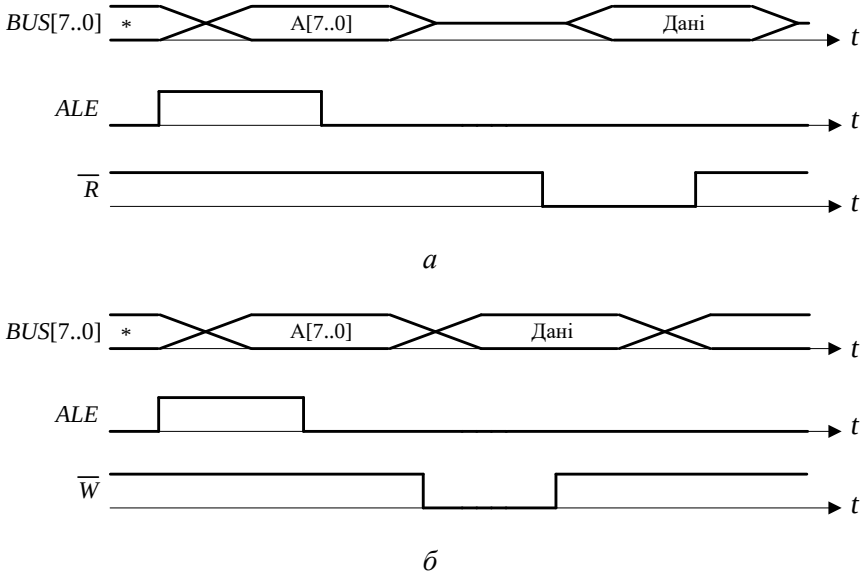


Рис.3.11. Часові діаграми обміну даними з зовнішньою пам'яттю даних: *а* – читання даних; *б* – та запису даних

Обмін інформацією між МК48 та зовнішньою пам'яттю (як з пам'яттю програми, так і з пам'яттю даних) здійснюється в синхронному режимі, тобто на обмін виділяється фіксований проміжок часу, причому сигнали квитирування (зворотного зв'язку), узгодженні за часом роботи МК48 і пам'яті, не передбачаються. В зв'язку з цим таке узгодження можливо реалізувати тільки за рахунок вибору тактуючої частоти роботи МК48.

Для реалізації більш складних програмно-апаратних способів доступу до зовнішньої пам'яті, її ємність може бути збільшена до необхідного об'єму за рахунок сторінкової організації, при цьому зовнішня пам'ять даних поділяється на сторінки по 256 байт в кожній, а зовнішня пам'ять програм – на сторінки по 4К байт. Для переключення між сторінками можна використовувати, вільні лінії портів *P*1 та *P*2.

Режим покрокового виконання програми

Режим покрокового виконання програми застосовується при налагодженні мікропроцесорних систем. Для його реалізації використовується вхід мікросхеми МК48 *SS* (рис. 3.1). Під час надходження на цей вхід сигналу низького рівня МК48 завершує

виконання чергової команди і зупиняється на етапі вибірки наступної команди. Підтвердженням зупинки є високий рівень сигналу *ALE*. В цьому стані на виходах МК48 *P*2[3..0] та *BUS* вже встановлена адреса наступної команди, яка може бути використана при аналізі стану системи в процесі налагодження. Для виходу МК48 з режиму зупинки необхідно подати сигнал високого рівня на вхід *SS*. При цьому на виході *ALE* встановлюється сигнал низького рівня і МК48 виконує чергову команду.

Схема реалізації режиму покрокового виконання програми ілюструється на рис. 3.12. Для вибору покрокового режиму використовується перемикач «Покроковий-автоматичний». *RS*-тригер застосовується для заглушення брязкоту контактів під час натискання кнопки «Крок». За натискання кнопки «Крок» запускається покроковий режим, при цьому *D*-тригер встановлюється в одиницю, тобто на вхід *SS* мікросхеми поступає додатний потенціал. Мікроконтролер завершує виконання чергової команди, виставляє адресу наступної команди і формує низький рівень сигналу *ALE*. В результаті цього *D*-тригер встановлюється в нульовий стан, що приводить до надходження низького рівня сигналу на вхід *SS*. Мікроконтролер переходить в режим зупинки до чергового натискання кнопки «Крок».

Часова діаграма сигналів при роботі МК48 в покроковому режимі показана на рис. 3.13.

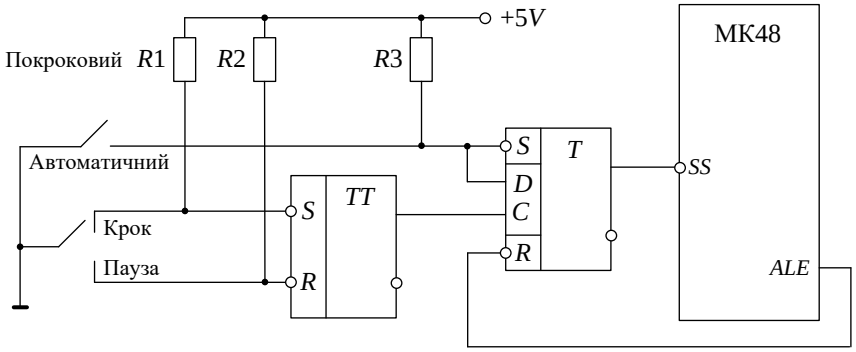


Рис. 3.12. Схема завдання режимів роботи мікроконтролера

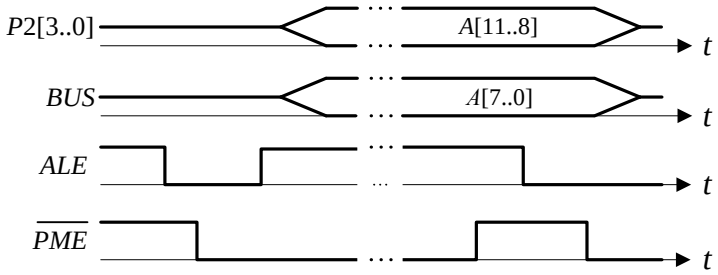


Рис. 3.13. Діаграма покрокового виконання команд

3.3.3. Підключення додаткових портів

Для збільшення кількості ліній зв'язку МК48 з об'єктом управління підключають додаткові чотирирозрядні порти *P*4, *P*5, *P*6, *P*7. Найбільш просто це здійснюється за використання спеціальної ІС КР580 ВР43, спосіб підключення якої до МК48 показаний на рис. 3.14. В цьому випадку забезпечується виконання всіх чотирьох команд роботи з додатковими портами – `MOVD A,Pp; MOVD Pp,A; ANLD Pp,A` та `ORLD Pp,A` (де $p = \overline{4,7}$), причому кожний вихід порту може бути налаштований як на введення так і на виведення інформації.

Команди передачі інформації між МК48 та додатковими портами виконуються за два цикли. В першому циклі на виходах *P*2[3..0] встановлюється управляюче слово, в другому циклі – через зазначені виходи здійснюється обмін інформацією між МК48 та одним з додаткових портів. Формат управляючого слова показаний на рис. 3.15. Для стробування даних в режимі підключення додаткових портів використовується сигнал *PROG*.

Відмітимо, що логічні операції І та АБО виконуються безпосередньо в ІС ВР43. Це необхідно враховувати при побудові додаткових портів з використанням інших апаратних засобів.

Часова діаграма роботи з додатковими портами *P*4, *P*5, *P*6, *P*7 показана на рис. 3.16.

Для розширення функціональних можливостей системи до МК48 можна підключати різні ІС, наприклад, адаптери КР580ВВ51, КР580ВВ55 тощо.

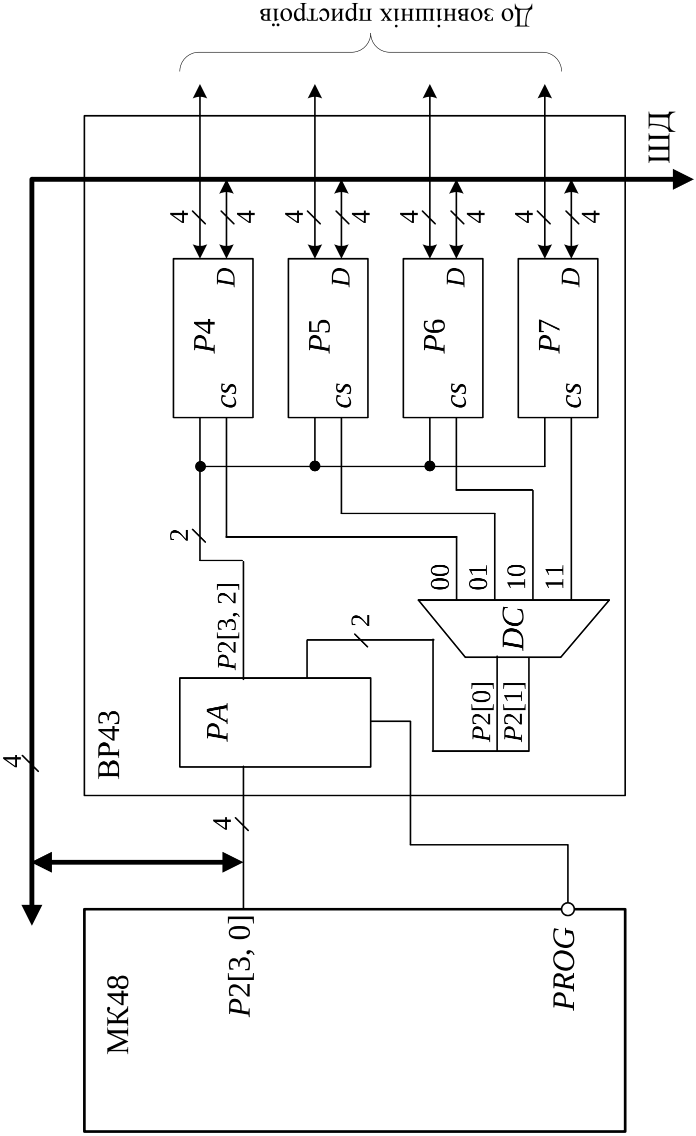


Рис. 3.14. Схема з’єднання виходів МК48 та ІС КР580ВР43

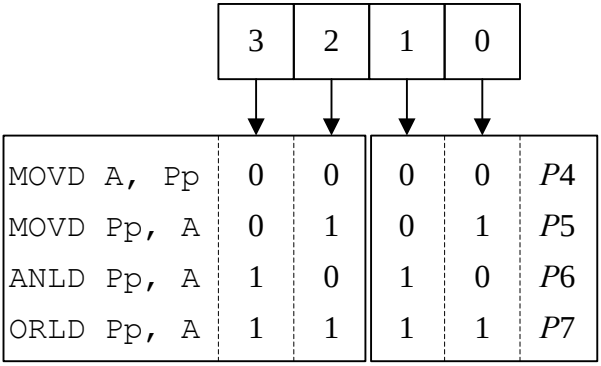


Рис. 3.15. Структура управляючого слова

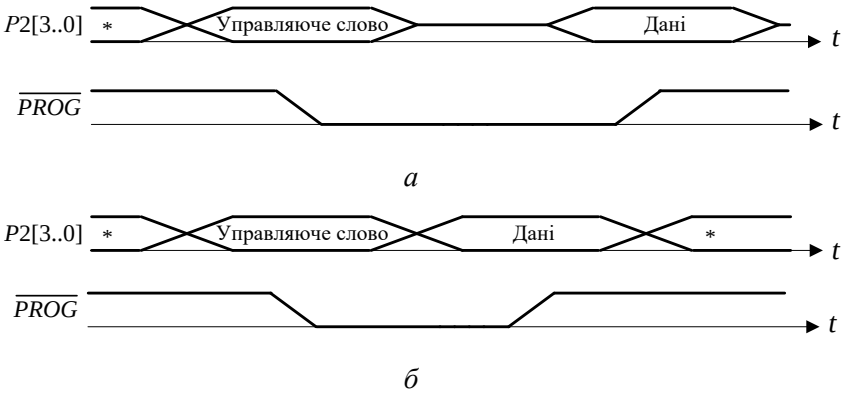


Рис. 3.16. Часова діаграма роботи з додатковими портами *P*4, *P*5, *P*6, *P*7: *а*- ввід даних, *б*- вивід даних

3.3.4 Підключення програмованого периферійного адаптера К580ВВ55

Основне призначення програмованих периферійних адаптерів (ППА) розробка програмованих пристроїв вводу/виводу для МПС.

Програмований периферійний адаптер 580ВВ55 (закордонний аналог ІС 8255А) виготовляється за *NMOS* технології. Може бути застосований у МПС з мікропроцесорами МК51, МК48, МП8086.

На рис. 3.17 показано умовне графічне позначення ППА К580ВВ55.

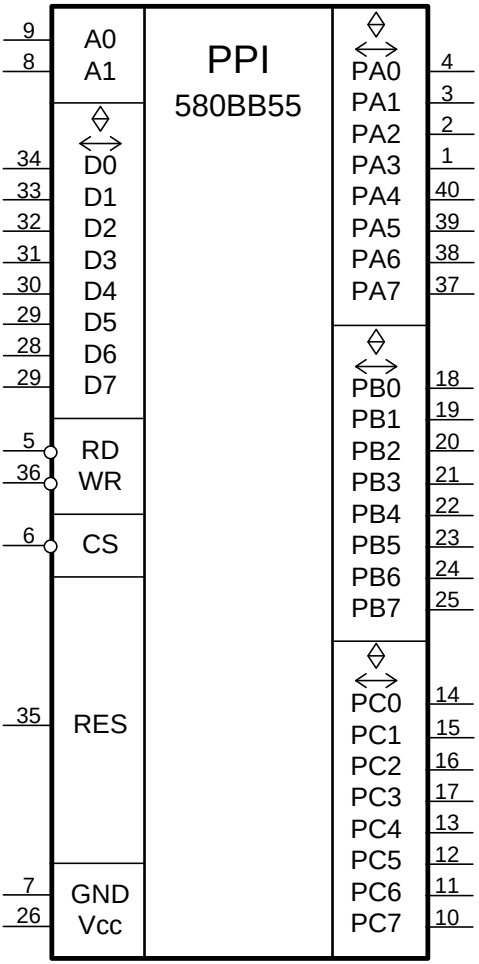


Рис. 3.17. Умовне графічне позначення програмованого периферійного адаптера К580ВВ55

Основні призначення виходів мікросхеми К580ВВ55:

*A*0, *A*1	–	сигнали двох молодших розрядів шини адреси
CS	–	вибір кристалу
RD	–	сигнал читання даних із ППА
WR	–	сигнал запису даних
RESET	–	сигнал установки початкового стану
*D*7 – *D*0	–	сигнали розрядів шини даних МК48

*PA*7 – *PA*0	–	входи/виходи порту *PA*
*PC*7 – *PC*0	–	входи/виходи порту *PC*
*PB*7 – *PB*0	–	входи/виходи порту *PB*
GND	–	загальний;
Vcc	–	напруга живлення +5В;

Структурна схема програмованого периферійного адаптера К580ВВ55 зображена на рис. 3.18.

Адаптер 580ВВ55 забезпечує ввід/вивід за трьома додатковими восьмирозрядними портами *PA, PB, PC*. Причому порт РС може застосовуватися в якості двох чотири розрядних портів *PCh* – старша тетрада порту *PC*, та *PCl* – молодша тетрада порту *PC*.

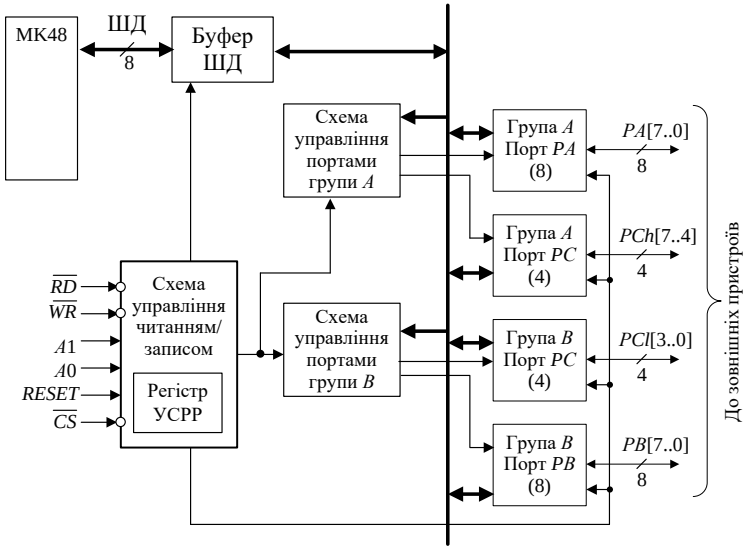


Рис. 3.18. Структурна схема програмованого периферійного адаптера К580ВВ55

До складу ППА входять наступні функціональні блоки.

- *буфер шини даних* (ШД) *D*7 – *D*0;
- *схема управління читанням/записом* даних в регістри ППА;
- *група А, порт PA* – порт вводу/виводу *PA* групи А;
- *група В, порт PB* – порт вводу/виводу *PB* групи В;
- *група С, порт PC* – порт вводу/виводу *PCh* групи А;
- *група В, порт PC* – порт вводу/виводу *PCl* групи В;
- схема управління портами групи А: порти *PA* та *PCh*;

– схема управління портами групи А: порти *PB* та *PCl*.

Схеми управління портами групи А та В містять регістр управління, що задає режими роботи портів.

Всі порти оснащенні буферними регістрами, через які здійснюється зв'язок між ППА і зовнішніми шинами.

Програмування режимів роботи і управління ІС здійснюється мікропроцесором за допомогою сигналів *D*7 – *D*0, *A*1, *A*0, $\overline{CS}$, $\overline{RD}$, $\overline{WR}$, RESET.

Адреси портів ППА входять до загального адресного простору зовнішньої пам'яті даних. Доступ до портів під час запису та читання здійснюється за застосування команд `MOVX A,@Rr`; `MOVX @Rr,A` (де, *r* = 1, 0).

Відмітимо, що по шині даних відбувається не тільки обмін даними, але і пересилання з МК48 в ППА управляючих слів, генерованих програмним забезпеченням процесора, а також передача в МК48 інформації про стан периферійного обладнання. Низький рівень сигналу на вході вибору кристалу *CS* дозволяє інформаційний зв'язок між ППА і МП48.

Обмін між МК48 і регістрами портів *PA, PB* і *PC* організовується під управлінням сигналів, що подаються на входи схеми управління читанням/записом. Функціональне призначення портів *PA, PB* і *PC* визначається управляючим словом, яке завантажується процесором у регістр управляючого слова режиму роботи УСРР (рис. 3.18). Сигнали на адресних входах *A*0 і *A*1 мікросхеми К580ВВ55 виробляють селекцію одного з трьох портів *PA, PB, PC* або регістру УСРР.

Восьмирозрядні порти програмованого периферійного адаптера можуть бути використані різними способами в залежності від характеристик конкретного устаткування.

Основні операції, які реалізуються ППА надані в табл. 3.2.

Управляюче слово режиму роботи (УСРР) використовується для налаштування режимів роботи ППА.

Програмований периферійний адаптер може знаходитися в таких основних режимах:

«0»	–	основний режим вводу/виводу інформації;
«1»	–	режим стробуючого вводу/виводу інформації;
«2»	–	режим двоспрямованої шини.

Таблиця 3.2. Основні операції, що виконує програмований периферійний адаптер К580ВВ55

Дії	**Сигнали управління**					**Операції**
	*A*1	*A*0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
Читання	0	0	0	1	0	РА $\rightarrow$ ШД (вивід в порт *PA*)
	0	1	0	1	0	РВ $\rightarrow$ ШД (вивід в порт *PB*)
	1	0	0	1	0	РС $\rightarrow$ ШД (вивід в порт *PC*)
Запис	0	0	1	0	0	ШД $\rightarrow$ РА
	0	1	1	0	0	ШД $\rightarrow$ РВ
	1	0	1	0	0	ШД $\rightarrow$ РС
	1	1	1	0	0	ШД $\rightarrow$ РУС
Відключення	*	*	*	*	*	ШД та порти відключені

Формат УСРР наведений на рис. 3.19.

Будь-який із перерахованих режимів може бути вибраний в ході виконання системної програми і встановлений завантаженням управляючого слова в регістр УСРР. Це дозволяє простими програмними засобами здійснювати реконфігурацію периферії МК48. Під час запису УСРР всі регістри пам'яті портів встановлюються в нуль, розряд D[7] – ознака встановлення режиму роботи, повинен бути встановлений у одиницю.

Якщо розряд D[7] = 0, управляюче слово не записується в регістр УСРР (на виходах *D*7 – *D*0 формується управляюче слово маніпуляції з бітами – УСМБ), в цьому випадку відбувається запис в один з тригерів регістру пам'яті порту *PC* значення 0 або 1, визначене розрядом D[0]. Розряди D[3] – D[1] задають номер тригера в регістрі пам'яті порту РС, в який завантажується значення вказане в розряді D[0] – встановлення/скидання біту порту *PC*. Формат УСМБ зображено на рис. 3.20.

Порт РС в цьому випадку є регістром пам'яті з розрядами, що адресуються. Розряд D[7] в управляючому слові, що надходить з шини даних адресує два внутрішніх регістра: регістр УСРР і регістр пам'яті порту РС. Такий спосіб передачі адресних сигналів внутрішніх вузлів по шині даних сприяє зменшенню числа зовнішніх виводів ІС.

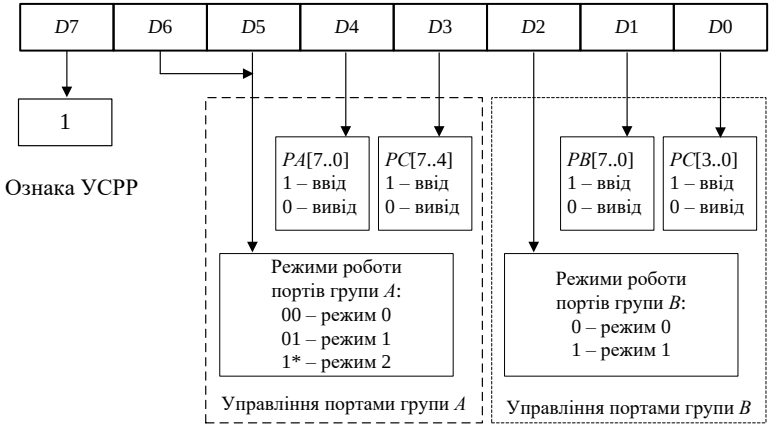


Рис. 3.19. Формат управляючого слова режиму роботи

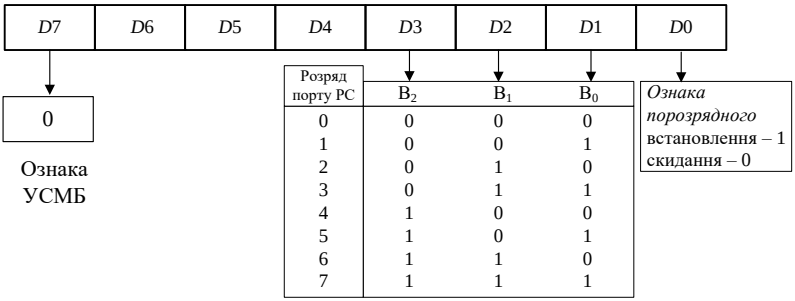


Рис. 3.20. Формат управляючого слова маніпуляції з бітами

Любий із розрядів порту *PC* може бути вибраний в якості тригера запиту переривання ЗПР або тригера фіксації дозволу переривання РПР. Програмний доступ до розрядів порту *PC* дає можливість розробити різноманітні процедури обробки переривань, пристосувавши їх до структури мікропроцесорної системи.

На рис. 3.21 приведена схема підключення програмованого периферійного адаптера до мікроконтролера МК48. За наявності одночасно зовнішньої пам'яті програм і зовнішньої пам'яті даних використовується один загальний зовнішній регістр адреси РА. Адреси портів *PA*, *PB*, *PC* та регістра управляючого слова РУС програмованого периферійного адаптера входять в загальній адресний простір однієї сторінки зовнішньої пам'яті даних. Для вибору ППА застосовується селектор адреси СА.

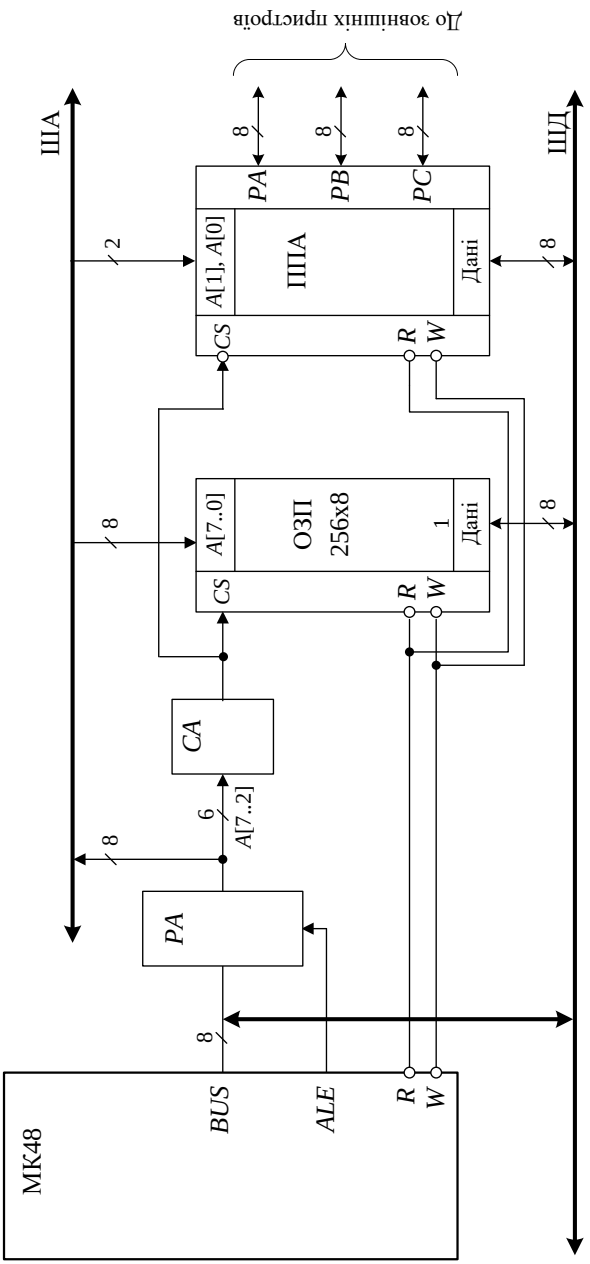


Рис. 3.21. Схема підключення програмованого периферійного адаптера К580ВВ55 до мікроконтролера МК48

3.3.5. Підключення програмованого зв'язкового адаптера К580ВВ51

В мікропроцесорних системах передача даних між функціональними пристроями здійснюється через системну магістраль. Дані передаються байтами в паралельному форматі. Передача даних в зовнішніх каналах зв'язку відбувається у вигляді послідовного потоку розрядів – в послідовному форматі. Для організації послідовних каналів зв'язку у МПС застосовуються додаткові пристрої – адаптери. Адаптери перетворюють дані, які знімаються з ШД з паралельного формату у послідовний формат, що придатний для передачі в відповідний канал зв'язку; а дані в послідовному форматі, що приймаються з каналу зв'язку, перетворює в паралельний формат, придатну для прийому в МПС.

Програмовані зв'язкові адаптери ПЗА є універсальними приймачами/передавачами, призначеними для роботи в ланцюгах послідовного синхронного/асинхронного обміну.

Програмований зв'язковий адаптер реалізований у вигляді ІС К580ВВ51 (закордонний аналог ІС 8251, та удосконалена ІС 8251А), умовне графічне позначення якого зображене на рис. 3.22.

Виготовляється ІС за *n*-МОП технології, входи/виходи сумісні з ТТЛ ІС.

Основні призначення виходів мікросхеми К580ВВ51:

$D7 - D0$	–	сигнали шини даних;
RESET	–	сигнал установки початкового стану, очікування УСРР;
CLK	–	сигнал синхронізації;
$C / \overline{D}$	–	управління/дані, адресний розряд $A0$ (див. табл. 3.5): 0 – читання/запис даних, 1 – читання/запис управляючих слів або слова стану;
$\overline{RD}$	–	читання;
$\overline{WR}$	–	запис;
$\overline{CS}$	–	вибір мікросхеми, сигнал з дешифратора адресних розрядів A[7..1];
$T \times D$	–	вихідні послідовні дані передавача;
$\overline{T \times C}$	–	тактовий сигнал передавача, частота якого визначає швидкість передачі за послідовним каналом зв'язку (біт/сек);
$T \times RDY$	–	готовність передавача, вказує МК48 на готовність

		буфера передавача прийнять байт даних для передачі у послідовний канал зв'язку;
$T \times E$	–	завершення передачі ($T \times EMPTY$): 1– відсутність у буфері даних передавача чергового символу для передачі за послідовним каналом зв'язку; 0 – отримання символу від МК48 при умові дозволу на передачу;
$R \times D$	–	вхідні послідовні дані приймача;
$\overline{R \times C}$	–	тактовий сигнал приймача, частота якого визначає швидкість передачі за послідовним каналом зв'язку (біт/сек) (зазвичай використовується один і той самий генератор тактових імпульсів для приймача і передавача, так що $\overline{T \times C} = R \times C$);
$R \times RDY$	–	готовність приймача;
SYNDET	–	*SYNDET* /*BRKDET* (виявлення синхронізації/ виявлення паузи): в синхронному режимі прийому є вихідним сигналом (*SYNDET*) під час завдання режиму внутрішньої синхронізації, є вхідним сигналом (*SYNDET*) під час завдання режиму зовнішньої синхронізації; в асинхронному режимі прийому (*BRKDET*) є вихідним сигналом;
GND	–	загальний;
Vcc	–	напруга живлення +5В;
$\overline{DTR}$	–	запит готовності передавача термінала ($\overline{DTR} = 0$), вихідний сигнал до модему застосовується для запиту його готовності передавати дані у ПЗА;
$\overline{DSR}$	–	готовність передавача термінала ($\overline{DSR} = 0$), вхідний сигнал, що поступає від модему у відповідь на сигнал $\overline{DTR}$, вказує на готовність модему передати дані у ПЗА;
$\overline{RTS}$	–	запит готовності приймача термінала ($\overline{RTS} = 0$); вихідний сигнал до модему застосовується для запиту його готовності прийняти дані від ПЗА;
$\overline{CTS}$	–	готовність приймача термінала, вхідний сигнал, що поступає від модему у відповідь на сигнал $\overline{RTS}$, застосовується для дозволу передачі даних від у ПЗА у модем;

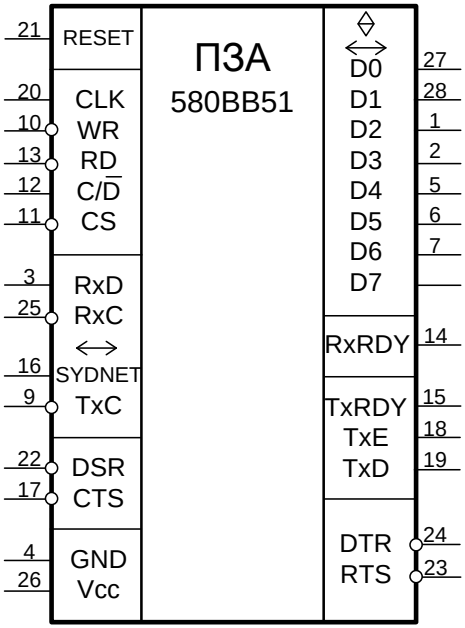


Рис. 3.22. Умовне графічне зображення програмованого зв'язкового адаптера К580ВВ51

Структурна схема ПЗА К580ВВ51 показана на рис.3.23.

До складу ПЗА входять:

– *буфер (приймач/передавач) шини даних* (ШД), що є восьмирозрядним регістром з трьома станами й використовується для обміну даними та управляючими словами;

– *блок управління записом/читанням*, що містить восьмирозрядні регістри: регістр управляючого слова режиму роботи (УСРР), регістр управляючого слова команди (УСК) та регістр управляючого слова статусу (УСС); блок записом/читанням приймає сигнали від МК48 та генерує внутрішні управляючи сигнали;

– *буфер передавача* (перетворення $P \to S$), який призначено для приймання символів у паралельному форматі P (*Parallel*) від МК48 та видачі послідовного потоку розрядів S (*Serial*) на вихід $T \times D$; буфер передавача (рис. 3.24) містить регістр вводу даних від МК48 та регістр зсуву *PI*/*SO* (*Parallel Input*/*Serial Output*), який перетворює дані з паралельного формату на послідовний;

– *схема управління передавачем*, застосовується для управління передачею;

– *буфер приймача* (перетворення $S \to P$), який виконує прийом послідовного потоку даних з входу $R \times D$ та передачу їх в МК48 в паралельному форматі; буфер приймача (рис. 3.24) містить регістр виводу даних в МК48 та регістр зсуву *SO*/*PI* (*Serial Output*/*Parallel Input*), який перетворює дані з послідовного формату на паралельний;

– *схема управління приймачем* застосовується для управління передачею, автоматично фіксує виявлені помилки парності, переповнення, кадру в управляючому слові стану;

– *схема управління модемом* (модулятор/демодулятор), що обробляє управляючи сигнали, призначені для зовнішнього пристрою.

Схема управління модемом

Сигнали управління модемом застосовуються для квитирування передачі послідовних даних між ПЗА та модемом. Квитирування – сигнал зворотного зв'язку, який призначений для інформування передавача про закінчення роботи приймача. Після чого передавач завершує операцію обміну та готовий до наступної передачі даних.

Устаткування, що приймає участь у передачі даних за послідовними калами зв'язку поділяють на два типи: термінальне устаткування – комп'ютери принтери, сканери і таке інше та зв'язувальне. Адаптер ПЗА входить до класу термінального устаткування. Типовим прикладом зв'язувального устаткування є модем, що з'єднує, наприклад, комп'ютер та телефонну лінію. Під час передачі даних за послідовними лініями зв'язку на обох кінцях лінії працюють модеми, що перетворюють цифрові сигнали на аналогові із застосуванням того чи іншого способу модуляції, під час прийому відбувається зворотне перетворювання – демодуляція.

Структурна схема дуплексного каналу зв'язку зображена на рис. 3.25. Для управління прийомом та передачею даних між ПЗА та модемом призначені пари сигналів $\overline{DTR}$, $\overline{DSR}$ та $\overline{RTS}$, $\overline{CTS}$.

Управління даними та режимами роботи ПЗА

Через буфер шини даних за сигналом запису $\overline{WR}$ МК48 записує у блок управління записом/читанням ПЗА дані або управляючі слова режиму роботи та команди (УСРР та УСК). За сигналом

читання $\overline{RD}$ МК48 зчитує з блоку управління записом/читанням ПЗА дані або управляюче слово стану (УСС).

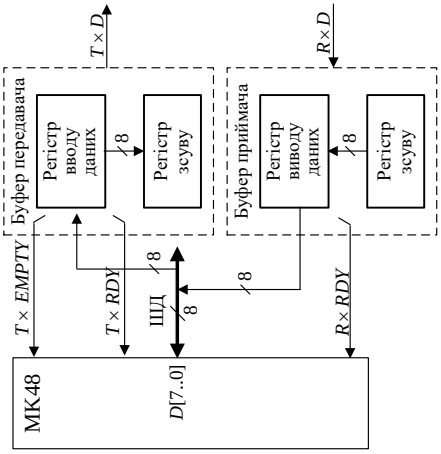


Рис. 3.24. Структурна схема буферів передавача та приймача

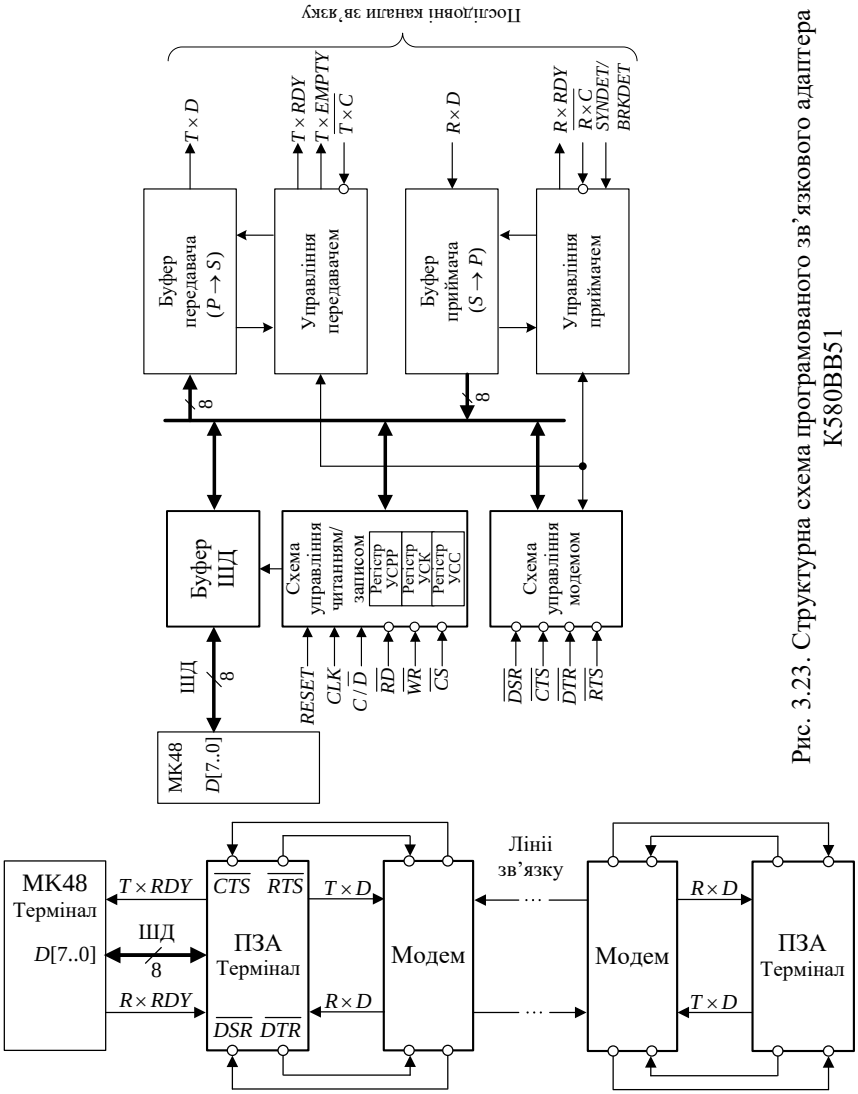


Рис. 3.23. Структурна схема програмованого зв'язкового адаптера К580ВВ51

Рис. 3.25. Структурна дуплексного схема каналу зв'язку

Сигнал на вході $C/\overline{D}$ вказує на тип інформації, що він приймає: управляюче слово, якщо $C/\overline{D}=1$, дані, якщо $C/\overline{D}=0$.

Будь які операції обміну даними можливі тільки в тому випадку, якщо на вході вибору мікросхеми $\overline{CS}$ встановлений нульовий сигнал.

Основні режими роботі ПЗА приводяться в табл. 3.3. Основні сигнали управління ($C/\overline{D}, \overline{RD}, \overline{WR}, \overline{CS}$) подаються на схему управління читанням/записом від МК48 та визначають тип оброблюваної інформації та напрямок передачі. Режими роботи ПЗА задається програмним шляхом після завантаження в нього управляючих слів УСРР та УСК з МК48.

Таблиця 3.3. Основні режими роботи програмованого зв'язкового адаптера К580ВВ51

Вхідні сигнали				**Операція (тип інформації)**
$C/\overline{D}$	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
0	0	1	0	ПЗА→ШД (ввід даних)
0	1	0	0	ШД→ПЗА (вивід даних)
1	0	1	0	ПЗА→ШД (читання УСС)
1	1	0	0	ШД→ПЗА (запис УСРР)
*	1	1	0	Немає операції (ШД у *Z*-стані)
*	*	*	1	Немає операції (ШД у *Z*-стані)

Таким чином, програмований зв'язковий адаптер оперує з управляючими словами наступних типів:

– управляюче слово режиму роботи;
– управляюче слово команди;
– управляюче слово стану.

Управляюче слово режиму роботи задає синхронний або асинхронний режим роботи, формат даних, швидкість прийому або передачі, необхідність контролю. Дані в регістр УСРР заносяться одразу після установки ПЗА в початковий стан програмно або за сигналом «*RESET*» та замінюються тільки під час зміни режиму.

Управляюче слово команди здійснює управління встановленим режимом обміну та може багаторазово змінюватись в процесі обміну даними під час управління його етапами.

В *управляючому слові стану* під час прийому даних за каналом зв'язку ПЗА фіксує виявлені помилки парності, переповнення, кадру.

В *асинхронному режимі* обмін даними відбувається із заздалегідь визначеною швидкістю. Приймач повинен бути узгоджений з передавачем за всіма параметрами формату сигналу, що передається, в тому числі і за часом передачі одного символу. В асинхронному режимі не вимагається сигнал зворотного зв'язку, який призначений для інформування передавача про закінчення роботи приймача.

Синхронний режим характеризується наявністю зворотного зв'язку приймача з передавачем, за яким передавач завершує операцію обміну та готовий до наступної передачі даних.

Під час асинхронного режиму управляюче слово команди завантажується одразу після управляючого слова режиму роботи, а під час синхронного режиму, перед УСК розташовуються один або два символи синхронізації.

В асинхронному режимі роботи формат кадру даних що передається за послідовними каналами зв'язку включає нульовий старт-біт, біти даних, контрольний біт та стоп-біти (рис. 3.26).

Високий рівень напруги називається пропуском, низький – маркером. Кадр розпочинається з передачі старт-біта (пропуску), далі передаються розряди даних, розпочинаючи з молодшого розряду, біт паритету *P* (контрольний розряд для визначення помилки за допомогою контролю на парність, або непарність) та один, півтора або два стоп-біта (маркери). Управляюче слово режиму роботи задає кількість бітів даних, кількість стоп-бітів, наявність біта контролю, та режим контролю.

Прийом кадру даних закінчується значенням стоп-біта. Якщо зчитаний пропуск (0) то в УСС фіксується *помилка кадру*. Якщо визначений маркер (1), приймач перетворює прийнятий послідовний код символу у паралельний і інформує МК48 про готовність даних. Якщо черговий прийнятий ПЗА з каналу зв'язку символ, не буде вчасно прочитаний МК48, він заміщується новим прийнятим символом і фіксується *помилка переповнювання*. Приймач здійснює перевірку прийнятого символу на парність або непарність. *Помилка паритету* також фіксується у УСС. Слід зазначити, що виявлення любої помилки не зупиняє роботу приймача.

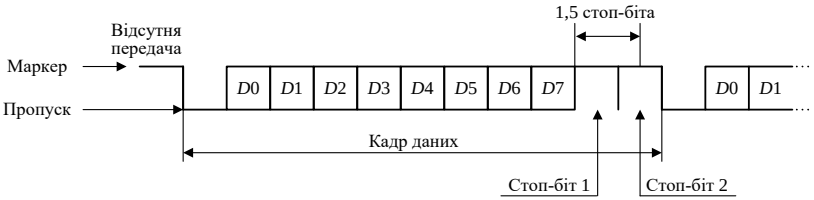


Рис. 3.26. Формат кадру даних під час асинхронної передача даних

Формат управляючого слова режиму роботи

Формат УСРР для асинхронного режиму обміну даними зображено на рис. 3.27.

Призначення розрядів управляючого слова режиму роботи для асинхронного режиму обміну даними наступне:

– розряди $D0$, $D1$ ($D0, D1 \neq 0$) – визначають три різновиди асинхронних режимів, що визначаються коефіцієнтом відношення частоти тактового сигналу передавача $\overline{T \times C}$ до частоти тактового сигналу приймача $\overline{R \times C}$ (1:1, 1:16 та 1:64);

– розряди $D3$, $D2$ – визначають кількість бітів даних, якщо довжина символу менш ніж вісім бітів, невикористаними будуть старші біти, які під час читання буферу приймача дорівнюватимуть нулю;

– розряди $D4$ – дозвіл або заборона встановлення розряду паритету;

– $D5$ – за умови $D4 = 1$ визначає режим контролю на парність/непарність;

– розряди $D6$, $D7$ – визначають кількість стоп-бітів для передачі.

Передавач автоматично додає до розрядів даних задану кількість старт-бітів, біти паритету (якщо $D4 = 1$) та задану кількість стоп бітів під час формування кадру даних відповідно до рис. 3.26.

Формат УСРР для синхронного режиму обміну даними зображено на рис. 3.28.

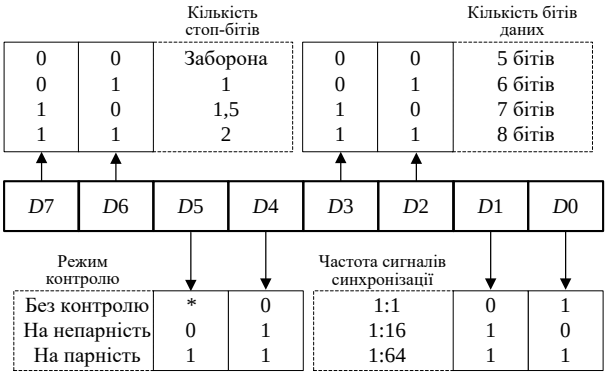


Рис. 3.27. Формат управляючого слова режиму роботи для асинхронного режиму

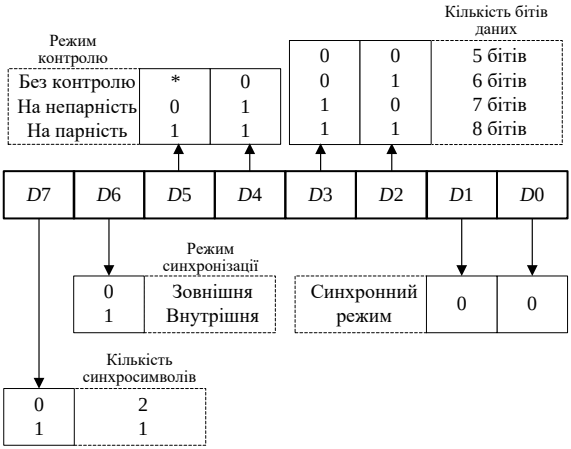


Рис. 3.28. Формат управляючого слова режиму роботи для синхронного режиму

Призначення розрядів управляючого слова режиму роботи для синхронного режиму обміну даними наступне:

– розряди $D0$, $D1$ – ($D0, D1 = 0$) завжди дорівнюють нулю для забезпечення синхронного режиму;

– розряд $D6$ – визначає режим зовнішньої ($D6 = 1$) або внутрішньої ($D6 = 0$) синхронізації;

– розряд $D7$ – визначає використання одного ($D7$ = 1) або двох ($D7$ = 0) символів синхронізації;

– розряди *D*2, *D*3, *D*4, *D*5 – мають теж саме значення, що й для асинхронного режиму.

В асинхронному режимі обміну даними у ПЗА завантажуються тільки регістр УСРР і регістр УСК. В синхронному режимі окрім регістрів УСРР та УСК завантажуються один або два символи синхронізації.

Формат управляючого слова команди

Управляюче слово команди подається після програмування режиму роботи та запису в регістри приймача синхросимволів при синхронному режимі роботи. Формат управляючого слова команди зображений на рис.3.29.

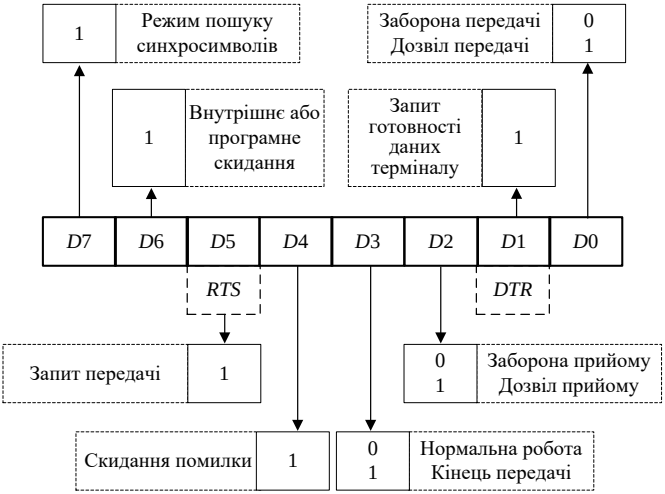


Рис. 3.29. Формат управляючого слова команди

Призначення розрядів управляючого слова команди наступне:

– розряд *D*0 – дозвіл передачі даних;

– розряд *D*1 = *DTR* – завдання значення $\overline{DTR} = 0$ сигналу запиту готовності даних терміналу, якщо *D*1 = 1;

– розряд *D*2 – дозвіл прийому;

– розряд *D*3 – завдання нормального режиму асинхронної передачі (*D*3 = 0) або паузи (*D*3 = 1) під час відсутності даних для передачі, вихід передавача $\overline{T \times C}$ встановлюється під час цього у стан 0;

– розряд *D*4 – скидання ознак помилок в управляючому слові стану;

– розряд *D*5 – стан входу *RTS*, завдання значення $\overline{RTS} = 0$ сигналу запиту передачі даних, якщо *D*5 = 1;

– розряд *D*6 – внутрішнє скидання ПЗА для переводу його в режим очікування управляючого слова режиму роботи;

– розряди *D*7 – дозвіл пошуку синхросимволів.

Формат управляючого слова статусу

Формат управляючого слова стану приводиться на рис. 3.30.

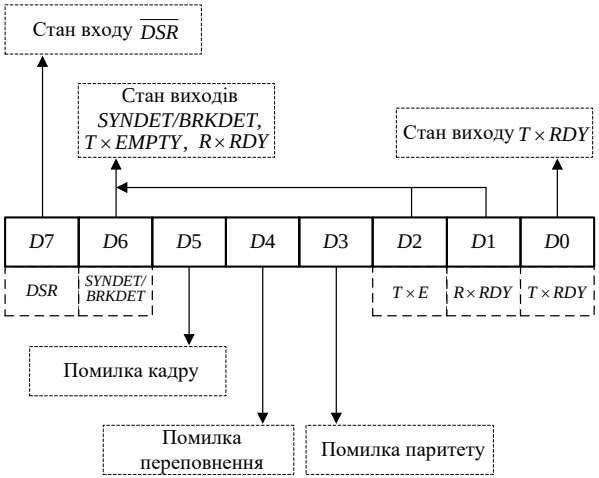


Рис. 3.30. Формат управляючого слова статусу

Призначення розрядів управляючого слова команди наступне:

– розряд *D*0 = $\overline{T \times RDY}$ – на відмінність від виходу $T \times RDY$ значення даного сигналу не переводиться в нуль сигналом $\overline{CTS} = 1$ і значенням розряду *D*0 УСРР;

– розряд *D*1 – стан виходу $T \times RDY$;

– розряд *D*2 – стан виходу $T \times EMPTY$;

– розряд *D*3 – помилка паритету;

– розряд *D*4 – помилка переповнення;

– розряд *D*5 – помилка кадру (застосовується тільки для асинхронного режиму);

– розряд *D*6 – стан виходу *SYNDET*/*BRKDET*;

– розряди *D*7 = *DTR* – стан виходу $\overline{DTR}$ (дані для терміналу підготовлені).

Широкі можливості використання адаптеру, які надаються форматом слова УСРР, обумовлюють використання ПЗА в мікропроцесорних системах різної складності і з різноманітними характеристиками інтерфейсу. Під час асинхронних передач є можливість виключити символи синхронізації, що дозволяє прикладній програмі малої мікропроцесорної системи використовувати переваги більш складної організації інтерфейсу і в той же час не бути обтяжливою непотрібним вантажем надлишкових команд.

Структурна схема підключення програмованого зв'язковий адаптера КР580ВВ51 до мікроконтролера МК48 приведена на рис. 3.31. Представлена мікропроцесорна система складається з зовнішньої пам'яті даних, програмованого зв'язкового адаптера, регістру адреси, дешифратора для вибору однієї з сторінок ЗПД. Оскільки адресний простір має об'єм більш ніж 256 Кб реалізується сторінкова організація ЗПД. На структурній схемі рис. 3.31 показаний спосіб підключення до МК48 n сторінок зовнішньої пам'яті даних з використанням k виходів порту $P1$ (де $k =]\log_2 n[$) і дешифратора *DC*.

Адреси портів ПЗА входять у загальний адресний простір зовнішньої пам'яті даних. Для уникнення перетину адрес загального адресного простору ЗПД та ПЗА можна застосовувати селектор адреси СА (аналогічно, як на схемі рис. 3.21).

Доступ до портів під час запису та читання здійснюється за застосування команд `MOVX A,@Rr`; `MOVX @Rr,A` (де, r = 1, 0).

Під час звернення до адрес у межах однієї сторінки ЗПД використовується одна команда `MOVX`. Переключення між сторінками потребує застосування додаткових команд.

В більшості випадків при підключенні до МК48 інтегральних схем адаптерів можна обійтись без додаткового устаткування (регістрів, дешифраторів і таке інше. Способи безпосереднього поєднання виходів МК48 з адаптерами КР580ВВ55 та КР580ВВ51 зображені на рис. 3.32.

На схемі, зображеній на рис. 3.32, *а* мається на увазі, що ЗПД складається тільки з регістрів ІС КР580ВВ55. Адреси регістрів задаються двома розрядами порту $P1$, тобто вміст регістрів $R0$ та $R1$ вибраного банку регістрів впливає на виконання команди `MOVX`.

Спосіб підключення КР580ВВ51 потребує декількох команд вводу-виводу для звернення до регістрів, однак система в цьому випадку містить зовнішню пам'ять даних та інше обладнання, пов'язане з шиною *BUS*.

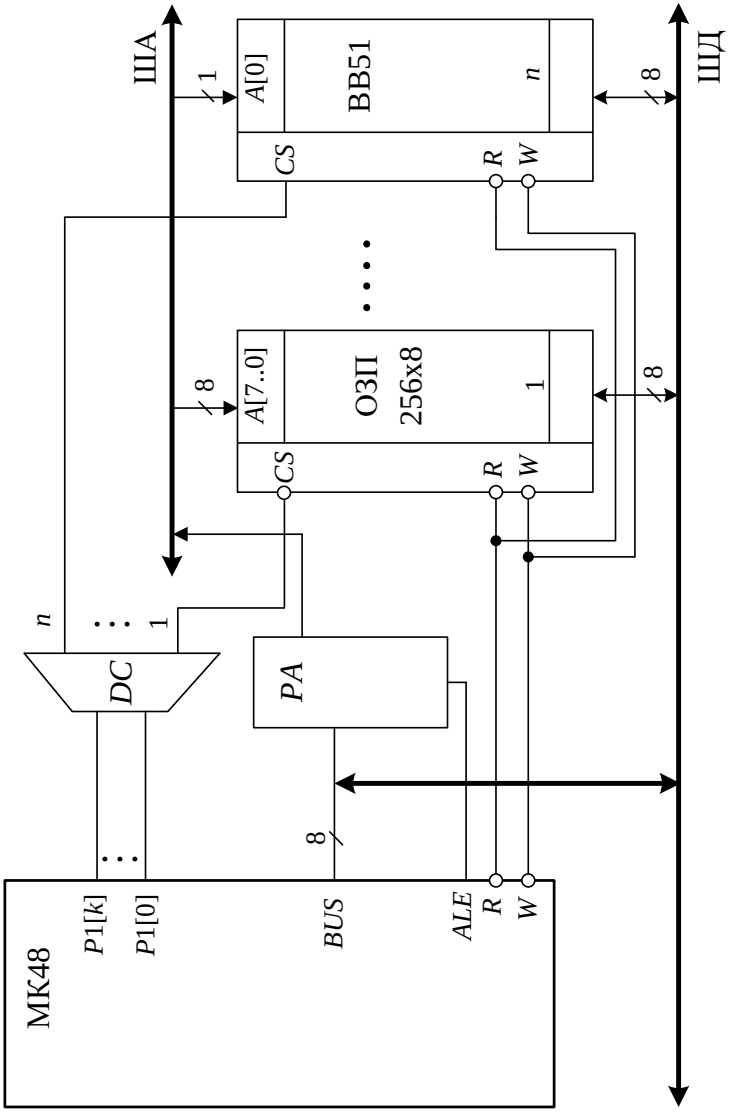


Рис. 3.31. Структурна схема підключення програмованого зв'язкового адаптера КР580ВВ51 до мікроконтролера МК48

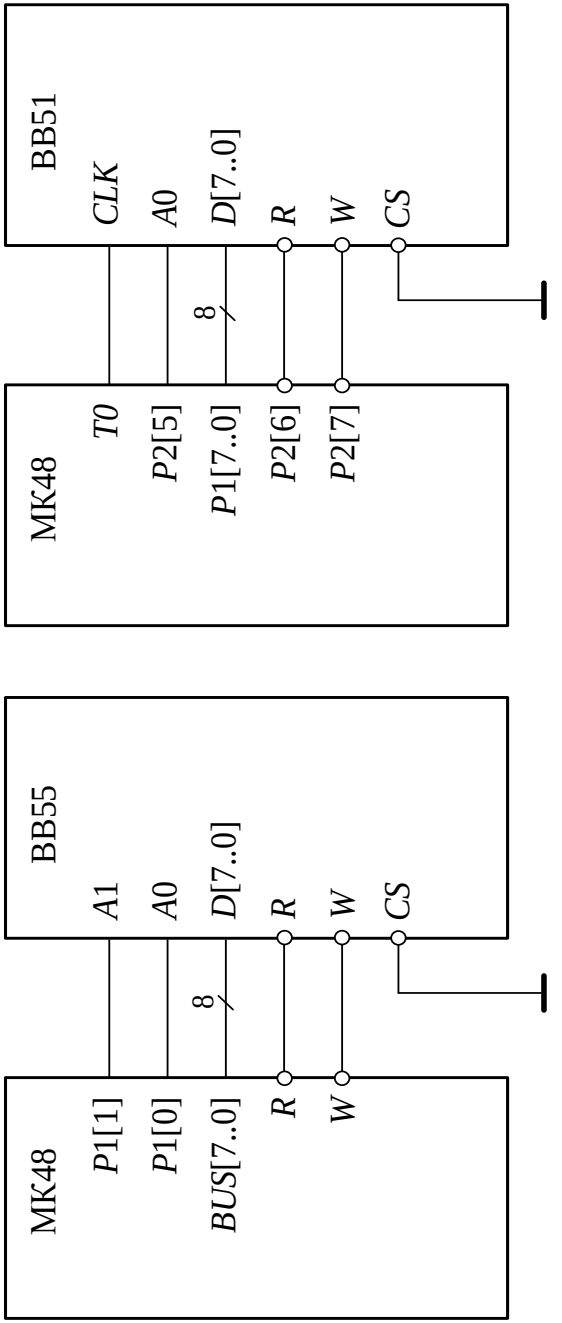


Рис. 3.32. Схеми підключення адаптерів до МК48: *а* – підключення інтегральної схеми МК580ВВ55; *б* – підключення інтегральної схеми МК580ВВ51

3.4. Система команд мікроконтролера КР1816ВЕ48

3.4.1. Формат команд

Всі команди МК48 мають формат 1 чи 2 байта і виконуються за один чи два машинні цикли. Кожний цикл виконується за 5 тактів. Частота синхронізації тактів складає $F/3$, а циклів – $F/15$. Наприклад, при заданій частоті F=6 МГц тривалість тактів і циклів складає 0,5 і 2,5 мкс відповідно. За два машинні цикли виконуються всі команди з безпосереднім операндом, команди введення-виведення, команди передачі управління і роботи з підпрограмами, а також команди пересилок `MOVX`, `MOVP`, `MOVP3`. Решта команд виконуються за один машинний цикл. В МК48 передбачена можливість суміщення виконання однієї команди і вибірки наступної, що може зменшити час виконання команди. Мікроконтролер оперує з командами чотирьох типів (рис. 3.33).

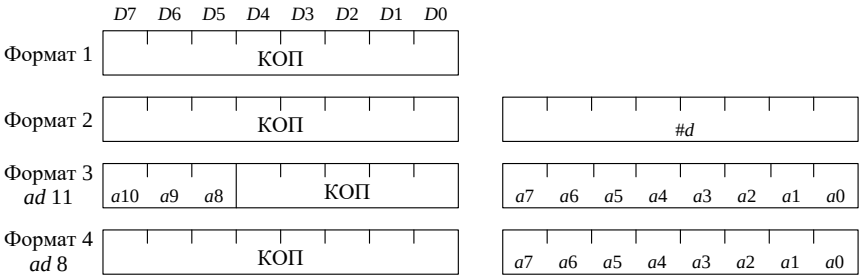


Рис. 3.33. Типи команд МК48

3.4.2. Система команд

У МК48 використовуються чотири способи адресації:
- пряма,
- безпосередня,
- непряма,
- неявна.

До переваг системи команд МК48 можна віднести: ефективний ввід/вивід, включаючи маскування і можливість управління окремими бітами портів; реалізацію розгалуження за значенням окремих бітів; обробку як двійкових, так і десяткових двійково-кодованих чисел.

Під час виконання команд використовуються ознаки, що формуються в регістрі слова стану програми *RSW*, і ознаки

користувача. Команди, в результаті виконання яких модифікуються ознаки, наведені в табл. 3.4. Функціональне призначення та способи формування ознак описані в розділі 3.2.2.

Таблиця 3.4. Формування ознак результату

Мнемоніка	**Ознаки**	**Мнемоніка**	**Ознаки**
ADD	*C* | *AC*	JTF	*TF* = 0
CLR C	*C* = 0	MOV PSW,A	*C* | *AC* | *F*0 | *BS*
CPL C	*C*	RETR	*C* | *AC* | *F*0 | *BS*
CLR F0	*F*0 = 0	RLC A	*C*
CLR F1	*F*1 = 0	RRC A	*C*
CPL F0	*F*0	SEL MB0; SEL MB1	*MB*
CPL F1	*F*1	SEL RB0; SEL RB1	*RB*
DA A	*C* | *AC*		

Для опису команд використовуються мнемокоди мови асемблера МК48. Під час запису символічного коду команд застосовуються наступні позначення:

A – акумулятор;
T – таймер;
r – номер регістру;
Rr – регістр з номером *r*;
b – номер біту;
p – номер порту вводу/виводу;
Pp – порт з номером *p*;
a – адреса;
d – безпосередньо операнд;
#d – безпосередньо операнд (восьмирозрядне двійкове число);
@Rr – операнд, що адресується непрямо через *Rr*;
@A – операнд, що адресується непрямо через A
Rr – робочий регістр.

В стовбці «Коментарі» наведений опис операцій, що виконуються під час виконання команди. Для запису операцій використовуються мова мікрооперацій із застосуванням, символічних імен і скорочень. Номери розрядів регістрів подаються у квадратних дужках [ ]. Операнд за непрямої адресації подається у дужках ( ). Наприклад, запис $A := (Rr)$ означає, що в акумуляторі A фіксується число, що зчитане з внутрішньої пам'яті даних за адресою, записаною в регістрі *Rr*. Для запису операндів часто

використовуються складені слова, що записані у вигляді двох слів, розділених крапкою. Наприклад, запис *PC*[11..8].*A* подає дванадцятирозрядне двійкове слово, вісім молодших розрядів якого є вмістом акумулятора *A*, а чотири старші розряди – вмістом бітів (11..8) лічильника команд *PC*.

Всю множину команд асемблеру МК48 можна розбити на чотири основні групи:

- команди пересилки даних,
- команди основної групи:
  - виконання арифметичних операцій;
  - виконання логічних операцій;
- команди передачі управління;
- команди управління режимами роботи.

Система команд наведена в табл. 3.5. Команди згруповані за функціональними ознаками.

Команди пересилки даних

На рис. 3.34 представлений граф, що ілюструє можливі операції пересилки даних в МК48 (табл. 3.5). Операнди розрізняються за місцем розташування і за способом адресації. В операціях пересилки приймають участь такі операнди: акумулятор, РЗП, *RSW*, таймер, порти вводу/виводу, безпосередньо операнд, ЗПД, РПД, ПП.

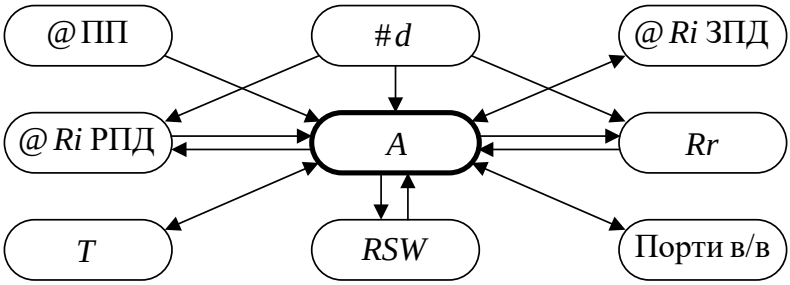


Рис. 3.34. Граф обміну даними у МК48

Всі операції пересилки даних в МК48 виконуються з застосуванням акумулятора *A*.

Під час пересилки даних між *A* та регістрами загального призначення РЗП банків регістрів, використовується пряма адресація, коли адреса операнда вміщується в тілі команди. Номер регістра, вміст якого пересилається в акумулятор вказується в трьох молодших бітах коду операції (рис. 3.33).

Таблиця 3.5. Система команд мікроконтролера КМ1816ВЕ48

Мнемоніка	Код команди	Коментарі
1	2	3
Основна група команд та команди пересилки даних		
Команди звернення до акумулятора		
CLR A	00100111	Встановлення вмісту акумулятора в нуль $A := 0$
CPL A	00110011	Інвертування вмісту A; $A := NOT\ A$
INC A	00010111	Інкремент вмісту A; $A := A + 1$
DEC A	00000111	Декремент вмісту A; $A := A - 1$
RR A	01110111	Циклічний зсув вмісту A вправо; $A[7] := A[0]$; $A[i] := A[i+1]$; $i = \overline{6,0}$
RL A	11100111	Циклічний зсув вмісту A вліво; $A[0] := A[7]$; $A[i] := A[i+1] := A[i]$; $i = \overline{6,0}$
RRC A	01100111	Циклічний зсув вмісту A з бітом переносу вправо; $A[7] := C$; $C := A[0]$; $A[i] := A[i+1]$; $i = \overline{6,0}$
RLC A	11110111	Циклічний зсув вмісту A з бітом переносу вліво; $A[0] := C$; $C := A[7]$; $A[i+1] := A[i]$; $i = \overline{6,0}$
SWAP A	01000111	Обмін тетрадами A; $A[7..4] \leftrightarrow A[3..0]$
DA A	01010111	Десяткова корекція вмісту A

Продовження табл. 3.5

1	2	3
`MOV A, Rr` ; $r=(7-0)$	`11111rrr`	Пересилка вмісту регістру в A; $A := Rr$
`MOV Rr, A` ; $r=(7-0)$	`10101rrr`	Пересилка вмісту A в регістр; $Rr := A$
`XCH A, Rr` ; $r=(7-0)$	`00101rrr`	Обмін вмісту A і регістру; $A \leftrightarrow Rr$
`ANL A, Rr` ; $r=(7-0)$	`01011rrr`	Логічне І вмісту A і регістру; $A := A\ AND\ Rr$
`ORL A, Rr` ; $r=(7-0)$	`01001rrr`	Логічне АБО вмісту A і регістру; $A := A\ OR\ Rr$
`XRL A, Rr` ; $r=(7-0)$	`11011rrr`	Виключне АБО вмісту A і регістру; $A := A\ XOR\ Rr$
`ADD A, Rr` ; $r=(7-0)$	`01101rrr`	Сума вмісту A і регістру; $A := A + Rr$
`ADDC A, Rr` ; $r=(7-0)$	`01111rrr`	Сума вмісту A, регістру і переносу C; $A := A + Rr + C$
`DEC Rr` ; $r=(7-0)$	`11001rrr`	Декремент вмісту регістру; $Rr := Rr - 1$
`INC Rr` ; $r=(7-0)$	`00011rrr`	Інкремент вмісту регістру; $Rr := Rr + 1$
Команди звернення до внутрішньої пам'яті даних		
`MOV A,@Rr` ; $r=0,1$	`1111000r`	Пересилка із внутрішньої пам'яті даних в A; $A := (Rr)$
`MOV @Rr,A` ; $r=0,1$	`1010000r`	Пересилка вмісту A до внутрішньої пам'яті даних; $(Rr) := A$
`XCH A,@Rr` ; $r=0,1$	`0010000r`	Обмін вмістом A і комірки внутрішньої пам'яті даних; $A \leftrightarrow Rr$
`XCHD A,@Rr` ; $r=0,1$	`0011000r`	Обмін молодшими тетрадами A і комірки внутрішньої пам'яті даних; $A[3..0] \leftrightarrow (Rr[3..0])$
`ANL A,@Rr` ; $r=0,1$	`0101000r`	Логічне І вмісту A і комірки внутрішньої пам'яті даних; $A := A\ AND\ (Rr)$

Продовження табл. 3.5

1	2	3
`ORL A,@Rr` ; $r=0,1$	`0100000r`	Логічне АБО вмісту A і комірки резидентної пам'яті даних; $A := A\,OR\,(Rr)$
`XRL A,@Rr` ; $r=0,1$	`1101000r`	Виключення АБО вмісту A і комірки резидентної пам'яті даних; $A := A\,XOR\,(Rr)$
`ADD A,@Rr` ; $r=0,1$	`0110000r`	Сума вмісту A і комірки резидентної пам'яті даних; $A := A + (Rr)$
`ADDC A,@Rr` ; $r=0,1$	`0111000r`	Сума вмісту A, комірки резидентної пам'яті даних і переносу C; $A := A + (Rr) + C$
`INC @Rr` ; $r=0,1$	`0001000r`	Інкремент комірки резидентної пам'яті даних; $(Rr) := (Rr) + 1$
Команди роботи з зовнішньою пам'яттю даних		
`MOVX A,@Rr` ; $r=0,1$	`1000000r`	Пересилка із ЗПД в A; $A := (Rr)$
`MOVX @Rr,A` ; $r=0,1$	`1001000r`	Пересилка вмісту A до ЗПД; $(Rr) := A$
Команди звернення до пам'яті програми		
`MOV Rr,#d` ; $r=(7-0)$	`10111rrr` `dddddddd`	Пересилка безпосереднього операнда до регістру; $(Rr) := d$
`MOV A,#d`	`00100011` `dddddddd`	Пересилка безпосередньої адреси до A $A := d$
`MOV @Rr,#d` ; $r=0,1$	`1011000r` `dddddddd`	Пересилка безпосереднього операнда до внутрішньої пам'яті даних $(Rr) := d$

Продовження табл. 3.5

1	2	3
ANL A,#d	01010011 dddddddd	Логічне І вмісту *A* з безпосереднім операндом; $A := A\ AND\ d$
ORL A,#d	01000011 dddddddd	Логічне АБО вмісту *A* з безпосереднім операндом; $A := A\ OR\ d$
XRL A,#d	11010011 dddddddd	Виключне АБО вмісту *A* з безпосереднім операндом; $A := A\ XOR\ d$
ADD A,#d	00000011 dddddddd	Сума вмісту *A* та безпосереднього операнду; $A := A + d$
ADDC A,#d	00010011 dddddddd	Сума вмісту *A*, безпосереднього операнду та переносу *C*; $A := A + d + C$
MOVP A,@A	10100011	Пересилка даних із поточної сторінки пам'яті програм до *A*; $A := (PC[11..8].A)$
MOVP3 A,@A	11100011	Пересилка даних із сторінки 3 пам'яті програм до *A*; $A := (0011.A)$
Команди звернення до регістру PSW		
MOV PSW,A	11010111	Пересилка вмісту *A* до регістру *PSW*; $PSW := A$
MOV A,PSW	11000111	Пересилка вмісту регістру *PSW* до *A*; $A := PSW$
MOV A,T	01000010	Пересилка вмісту *TCNT* в *A*; $A := TCNT$
MOV T,A	01100010	Пересилка вмісту *A* в *TCNT*; $TCNT := A$
Команди встановлення ознак		
CLR C	10010111	Встановлення в нуль ознаки *C*; $C := 0$

Продовження табл. 3.5

1	2	3
`CPL C`	`10100111`	Інвертування ознаки C; $C := NOT\ C$
`CLR F0`	`10000101`	Встановлення в нуль ознаки $F0$; $F0 := 0$
`CLR F1`	`10100101`	Встановлення в нуль ознаки $F1$; $F1 := 0$
`CPL F0`	`10010101`	Інвертування ознаки $F0$; $F0 := NOT\ F0$
`CPL F1`	`10110101`	Інвертування ознаки $F1$; $F1 := NOT\ F1$
Команди звернення до портів P1 і P2		
`ANL Pp,#d` ; $p = 1, 2$	`100110pp` `dddddddd`	Логічне І порту $P1(P2)$ з безпосереднім операндом; $Pp := Pp\ AND\ d$
`ORL Pp,#d` ; $p = 1, 2$	`100010pp` `dddddddd`	Логічне АБО порту $P1(P2)$ з безпосереднім операндом; $Pp := Pp\ OR\ d$
`IN A,Pp` ; $p = 1, 2$	`000010pp`	Введення даних із порту $P1(P2)$ в A; $A := Pp$
`OUTL Pp,A` ; $p = 1, 2$	`001110pp`	Виведення вмісту A в порт $P1(P2)$ $Pp := A$
Команди звернення до портів P4, P5, P6, P7		
`ANLD Pp,A` ; $p = (7-4)$	`100111pp`	Логічне І порту $P4(P5, P6, P7)$ з A; $Pp := Pp\ AND\ A[3..0]$
`ORLD Pp,A` ; $p = (7-4)$	`100011pp`	Логічне АБО порту $P4(P5, P6, P7)$ з A; $Pp := Pp\ OR\ A[3..0]$
`MOVD A,Pp` ; $p = (7-4)$	`000011pp`	Ввід із порту $P4(P5, P6, P7)$ в A; $A[7..4] := 0$; $A[3..0] := Pp$
`MOVD Pp,A` ; $p = (7-4)$	`001111pp`	Вивід молодшої тетради із A в порт $P4(P5, P6, P7)$; $Pp := A[3..0]$

Продовження табл. 3.5

1	2	3
Команди звернення до порту BUS		
ANL BUS,#d	10011000 dddddddd	Логічне І порту *BUS* з безпосереднім операндом; $BUS := BUS\ AND\ d$
ORL BUS,#d	10001000 dddddddd	Логічне АБО порту *BUS* з безпосереднім операндом; $BUS := BUS\ \ OR\ d$
INS A,BUS	00001000	Ввід даних із порту *BUS* в *A*; $A := BUS$
OUTL BUS,A	00000010	Вивід вмісту *A* в порт *BUS*; $BUS := A$
Команди передачі управління		
JMP a	aaa00100 aaaaaaaa	Безумовний перехід $PC[10..0] := a[10..0]$; $PC[11] := MB$
JMPP @A	10110011	Безумовний перехід в межах поточній сторінці; $PC[7..0] := (A)$
JC a	11110110 aaaaaaaa	Перехід, якщо $C = 1$, то $PC[7-0] := a$ інакше $PC := PC + 2$
JNC a	11100110 aaaaaaaa	Перехід, якщо $C = 0$
DJNZ Rr,a	11101rrr aaaaaaaa	Декремент вмісту регістру і перехід, якщо вміст регістру не дорівнює нулю
JZ a	11000110 aaaaaaaa	Перехід, якщо вміст *A* дорівнює нулю
JNZ a	10010110 aaaaaaaa	Перехід, якщо вміст *A* не дорівнює нулю

Продовження табл. 3.5

1	2	3
JF0 a	10110110 aaaaaaaa	Перехід, якщо $F0=1$
JF1 a	01110110 aaaaaaaa	Перехід, якщо $F1=1$
JT0 a	00110110 aaaaaaaa	Перехід, якщо $T0=1$
JNT0 a	00100110 aaaaaaaa	Перехід, якщо $T0=0$
JT1 a	01010110 aaaaaaaa	Перехід, якщо $T1=1$
JNT1 a	01000110 aaaaaaaa	Перехід, якщо $T1=0$
JTF a	00010110 aaaaaaaa	Перехід, якщо $TF=1$
JNI a	10000110 aaaaaaaa	Перехід, якщо $INT=0$
JBb a	bbb10010 aaaaaaaa	Перехід, якщо розряд *Bb* акумулятора встановлений в одиницю, де $b=(7-0)$
CALL a	aaa10100 aaaaaaaa	Виклик підпрограми; $SP:=SP+1$; $(SP):=PSW[7..4]$; $PC[11]:=MB$; $PC[10..0]:=a[10..0]$
RET	10000011	Повернення із підпрограми; $SP:=SP-1$; $PC:=(SP[11..0])$

Продовження табл. 3.5

1	2	3
RETR	10010011	Повернення із підпрограми з встановленням стану; $SP := SP - 1$; $PC := SP[11..0]$; $PSW[7..4] := (SP[15..12])$
Команди управління режимами роботи		
ENTO CLK	01110101	Дозвіл видачі імпульсів синхронізації на $T0$
SEL MBO	11100101	Вибір нульового банку пам'яті програм; $MB := 0$
SEL MB1	11110101	Вибір першого банку пам'яті програм; $MB := 1$
SEL RBO	11000101	Вибір нульового банку регістрів пам'яті даних; $RB := 0$
SEL RB1	11010101	Вибір першого банку регістрів пам'яті даних; $RB := 1$
NOP	00000000	Немає операції
EN I	00000101	Дозвіл зовнішніх переривань
DIS I	00010101	Заборона зовнішніх переривань
EN TCNTI	00100101	Дозвіл переривань від таймера/лічильника
DIS TCNTI	00110101	Заборона переривань від таймера/лічильника
STRT T	01010101	Запускання таймера/лічильника в режимі таймера
STRT CNT	01000101	Запускання таймера/лічильника в режимі лічильника
STOP TCNT	01100101	Зупинка таймера/лічильника

Обмін даними між *A* та комірками РПД або ЗПД здійснюється з використанням непрямої адресації. При цьому, покажчики адреси розміщуються в регістрах *R*0 або *R*1 вибраного банку регістрів.

За неявної адресації у коді команди неявно указується один з операндів. Найчастіше таким операндом є акумулятор. За безпосередньої адресації у тілі команди в якості другого байту вказується безпосередній операнд (константа), який пересилається за місцем призначення, визначеним першим операндом.

До пам'яті програм здійснюється доступ лише у напряму читання даних.

Всі команди, окрім `MOV PSW,A`, не впливають на встановлення ознак.

Більшість команд виконує пересилку восьмибітних даних. Декілька команд оперують з чотирибітними операндами (тетрадами) і застосовуються для звернення до чотирибітних портів вводу/виводу *P*4, *P*5, *P*6, *P*7.

В МК48 передача даних відбувається в двох режимах: пересилки (завантаження) и обміну. Під час пересилки дані передаються від джерела до приймача, при цьому джерело даних не змінює свого вмісту. Обмін припускає одночасну передачу даних в обох напрямах при цьому змінюються значення обох операндів, що приймають участь в обміні.

Команди пересилки даних всередині МК48 виконуються за один машинний цикл, а обмін даними з ЗПД потребує двох машинних циклів.

Приклади команд:

```
MOV   A,Rr     ; (A) := Rr , r = (7 – 0); пряма адресація.
MOV   A,#d     ; A := d ; безпосередня адресація.
MOV   A,#05    ; A := 05 ; безпосередня адресація.
MOV   Rr,#d    ; (Rr) := d , r = (7 – 0).
MOV   A,PSW    ; (A) := PSW .
MOV   A,T      ; (A) := T .
MOV   A,@R0    ; (A) := ((Rr)) , r = 0,1; непряма адресація.
XCH   A,Rr     ; (A) ↔ (Rr) , r = (7 – 0).
XCH   A,#Rr    ; (A) ↔ (Rr) , r = (7 – 0).
```

Команди основної групи

До команд основної групи належать команди виконання арифметичних та логічних операцій. В МК48 виконуються наступні операції над восьмибітними цілими двійковими числами без знаку: двійкове додавання, двійкове додавання з урахуванням переносу, операції десяткової корекції, інкременту, декременту, зсуву, кон'юнкції, диз'юнкції тощо (табл. 3.5).

Дві логічні команди скидання `CLR` та інверсія `CPL` дозволяють виконувати операції з бітами.

Під час додавання застосовується неявна адресація джерела першого операнду і місця розташування результату, в якості яких використовується акумулятор. До вмісту акумулятора можна додавати вміст регістру РЗП, константу, вміст комірки РПД. Під час підсумовування формується ознака переносу *C*, що фіксується у відповідному розряді регістру *RSW* (рис. 3.3). Команда підсумовування `ADDC` з урахуванням переносу дозволяє виконувати підсумовування багатобайтних чисел.

В МК48 відсутня безпосередньо операція віднімання, при цьому віднімання реалізується за наступної послідовності дій: отримання додаткового коду другого операнду, додавання його до вмісту А, де зберігається перший операнд, та подання результату в доповнювальному коді.

Складні арифметичні операції ділення, множення, піднесення до ступеня, тощо виконуються за підпрограмами.

Команди виконання арифметичних операцій змінюють відповідні ознаки в регістрі слова стану програми *RSW* (табл. 3.4).

Приклади команд:

`ADD A,Rr` ; $(A) := (A) + (Rr)$, $r = (7-0)$.

`ADD A,#d` ; $(A) := (A) + d$.

`ADD A,@Rr` ; $(A) := (A) + ((Rr))$, $r = 0,1$.

`ADDC A,Rr` ; $(A) := (A) + (Rr) + C$, $r = (7-0)$.

`INC A` ; $(A) := (A) + 1$.

`RL A` ; $(A_{i+1}) := (A_i)$, $(A_0) := (A_7)$, $i = (7-0)$.

`RLC A` ; $(A_{i+1}) := (A_i)$, $(A_0) := (C)$, $(C) := (A_7)$, $i = (7-0)$.

`RR A` ; $(A_i) := (A_{i+1})$, $(A_7) := (A_0)$, $i = (7-0)$.

`RRC A` ; $(A_i) := (A_{i+1})$, $(A_7) := (C)$, $(C) := (A_0)$, $i = (7-0)$.

```
ANL   A,Rr;   ; (A) := (A) & (Rr), r = (7 – 0).
ORL   A,@Rr;  ; (A) := (A) ∨ (Rr), r = 0,1.
XRL   A,#d;   ; (A) := (A) ⊕ d.
CLR   A;      ; (A) := 0.
CPL   A;      ; (A) := (Ā).
CLR   C;      ; (C) := 0.
CPL   F1;     ; (F1) := (F̄1).
```

Принцип виконання зсувів командами `RL`, `RLC`, `RR`, `RRC` ілюструє схема, що показана на рис. 3.35.

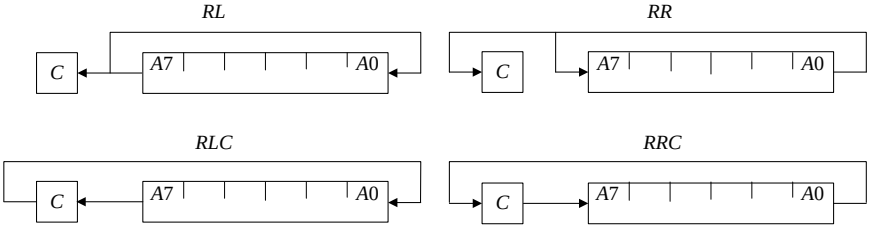


Рис. 3.35. Схема виконання операцій зсуву

Команди передачі управління

В групу команд передачі управління входять дві команди безумовного переходу, команди умовних переходів, команда виклику підпрограми і дві команди повернення з підпрограм.

Виконання команд передачі управління і роботи з підпрограмами має свої особливості і залежить від розподілу програм і даних за банками і сторінками пам'яті.

Більшість команд передачі управління застосовують *пряму адресацію*, при цьому в тілі команди вказується адреса переходу.

Команди *безумовного переходу* `JMP` і `CALL` вміщують одинадцятибітну адресу переходу і дозволяють здійснити перехід до будь-якої комірки ЗПП тільки в межах одного банку пам'яті програм (2048 адресів). Номер банку пам'яті програм визначається ознакою *MB*, значення якого копіюється у старший біт лічильника команд (*PC*11) під час виконання команд `JMP` або `CALL`.

Перехід до іншого банку пам'яті програм можливий тільки за використання команд із групи команд управління режимами роботи `SEL MB0` та `SEL MB1`. До тих пір поки не буде обрано інший блок

пам'яті, всі переходи здійснюються в межах поточного банку пам'яті.

Інші команди є *командами умовних переходів* (окрім команд повернення), вміщують восьмибітну адресу переходу і дозволяють здійснювати передачу управління лише в межах поточної сторінки (256 байт). Така адресація називається *короткою прямою адресацією*, вона обумовлює певні обмеження на розміщення програм у пам'яті.

За застосування команд умовних переходів ознаки, що аналізуються, за виключенням *C* і *F*0 не фіксуються в спеціальних регістрах ознак, а подаються короткотривалими значеннями сигналів на виходах АЛП або відповідних входах МК48.

За застосування команд умовних переходів можна перевіряти не тільки внутрішні ознаки, але і деякі сигнали на зовнішніх входах МК48. Це дозволяє виконувати ефективне розгалуження в програмах без попереднього вводу і аналізу.

Непряма адресація реалізована командою `JMPP`, яка здійснює перехід за адресою, що зберігається у вказаній комірці ПП. Покажчиком адреси є акумулятор. Перехід також здійснюється у межах однієї сторінки ПП.

Для організації програмних циклів використовується команда `DINZ` (декремент регістру і перехід, якщо не нуль), яка дозволяє використовувати будь-який із восьми робочих регістрів в якості лічильника і здійснює перехід в межах поточної сторінки пам'яті програм.

Для роботи з підпрограмами використовуються команди `CALL` та `RET`. Команда `CALL`, дозволяє звернутись до любої комірки поточного банку ПП. Під час виклику підпрограми в стеку запам'ятовується адреса повернення і частина регістру *RSW*. Кількість вкладань обмежується глибиною стеку (16 байтів) и не повинна перевищувати восьми. Під час повернення з підпрограми в лічильнику команд поновлюється адреса повернення. Команда `RETR` використовується для повернення з підпрограми обробки переривань, окрім адреси повернення поновлює вміст регістру *RSW* і встановлює дозвіл переривання від даного джерела.

Приклади команд:

```
JMP   a     ; (PC[10..8]) := A[10..8],
            ; (PC[7..0]) := A[7..0], (PS11) := (MB).
```

```
DJNZ  Rr,a ; (Rr) := (Rr) − 1; якщо (Rr) ≠ 0, то
           ; (PC[7..0]) := a , в іншому випадку (PC) := (PC) + 2 .
JC    a    ; якщо C = 1, то (PC0−7) := a , в іншому випадку
           ; (PC) := (PC) + 2 .
JZ    a    ; якщо (A) = 0, то (PC0−7) := a , в іншому випадку
           ; (PC) := (PC) + 2 .
JBb   a    ; якщо (Bb) = 1, то (PC0−7) := a , в іншому випадку
           ; (PC) := (PC) + 2 .
CALL  a    ; виклик підпрограми.
RETR       ; повернення з підпрограми з відновленням PSW.
```

Команди управління режимами роботи

В групу команд управління режимами роботи входять команди управління таймером/лічильником, перериваннями и ознаками переключення банків регістрів и банків пам'яті програм.

Вище були розглянуті команди обміну інформацією між таймером і акумулятором (`MOV A,T` и `MOV T,A`), за виконання яких може бути прочитано вміст таймера після зупинки підрахунку або безпосередньо під час підрахунку («на льоту»), а також за необхідністю перезавантажено вміст таймера. Окрім цього в МК48 виконуються спеціальні команди управління режимом роботи таймера.

Таймер, залежно від застосованої команди, може бути використаний як лічильник тактів від внутрішнього джерела сигналів або як лічильник подій від зовнішнього джерела сигналів.

Система команд МК48 має в своєму розпорядженні засоби дозволу або заборони переривання від таймера. Спеціальною командою `ENT0` на вивід *T*0 дозволяється передача імпульсів з частотою тактового синхросигналу, діленою на три. Видача цього сигналу може бути відключена тільки сигналом загального скидання. Синхросигнал на виході *T*0 використовується для загальної синхронізації зовнішніх пристроїв, узгоджених з МК48 за частотою роботи.

Приклади команд:

```
MOV   T,A     ; завантаження таймеру.
STRT  T       ; запуск таймеру.
STRT  CNT     ; запуск лічильника.
```

```
EN      TCNTI    ; дозвіл переривання від таймеру.
DIS     TCNTI    ; заборона переривань від таймеру.
```

3.5. Розробка програм обробки даних

3.5.1. Виконання команд передачі даних

***Приклад 3.1*:** Завантажити в регістри *R*5 та *R*6 нульового банку регістрів число 1*FFFh*:

```
SEL   RB0       ; вибір банку регістрів
MOV   R5,#1Fh   ; запис в R5 числа 1Fh
MOV   R6,#FFh   ; запис в R6 числа FFh
```

***Приклад 3.2*:** Завантажити в комірки РПД з адресами 55 і 56 число 20*A*0*h*:

```
MOV   R0,#55h   ; завантаження в R0 вказівника адреси
MOV   @R0,#20h  ; запис в комірку 55 числа 20h
INC   R0        ; +1 до вказівника адреси
MOV   @R0,#A0h  ; запис в комірку 56 числа A0h
```

***Приклад 3.3*:** Завантажити в таймер число $(-5)_{ДК}$ без втрати вмісту акумулятора:

```
XCH   A,R2      ; обмін R2 і акумулятора
MOV   A,#FBh    ; завантаження в А числа (– 5)ДК
MOV   T,A       ; пересилка в таймер
XCH   A,R2      ; обмін R2 та A
```

***Приклад 3.4*:** Переслати вміст регістрів *R*2, *R*3, *R*7 першого банку регістрів в ЗПД, починаючи з комірки *C*0*h* другої сторінки пам'яті

```
SEL     RB1     ; вибір першого банку регістрів
ANL     P1,#0   ; встановлення в нуль P1
ORL     P1,#4h  ; вибір другої сторінки ЗПД
MOV     A,R2    ;A:=R2
```

```
MOV      R0,#C0h  ; R0 := C0h адреса комірки
MOV X    @R0,A    ; пересилка R2 в комірку з адресою C0h
INC      R0       ; +1 до вказівника адреси
MOV      A,R3     ; A := R3
MOV X    @R0,A    ; пересилка R3 в комірку з адресою C1h
INC      R0       ; +1 до вказівника адреси
MOV      A,R7     ; A := R7
MOV X    @R0,A    ; пересилка R7 в комірку з адресою C2h
```

3.5.2. Виконання команд арифметичних та логічних операцій

***Приклад 3.5*:** Виконати додавання вмісту регістру *R*5 нульового банку регістрів і вмісту комірки РПД з адресою 56:

```
SEL   RB0      ; вибір першого банку регістрів
MOV   R1,#56   ; завантаження вказівника адреси
MOV   A,R5     ; A := R5
MOV   @R1,A    ; пересилка R5 в комірку з адресою 56
```

***Приклад 3.6*:** Виконати додавання багатобайтних чисел, розміщених в РПД. Регістри *R*1 та *R*0 першого банку регістрів вказують початкові адреси, розміщення в РПД молодших байтів вихідних аргументів.

```
; додавання Z = W + Y
; R0 – початкова адреса W
; R1 – початкова адреса Y
; R2 – довжина W та Y
        SEL     RB1       ; вибір першого банку регістрів
        CLR     C         ; скидання прапору переносу
LOOP:   MOV     A,@R0     ; завантаження поточного байту
        ADDC    A,@R1     ; додавання (W+Y)
        DA      A         ; корекція
        MOV     @R0,A     ; розміщення чергового байту
                          ; результату
        INC     R0        ; +1 до покажчика адреси
        INC     R1        ; +1 до покажчика адреси
```

```
        DYNZ    R2,LOOP  ; декремент R2 та повернення на
                         ; початок циклу поки R2
                         ; не дорівнюватиме нулю.
```

Приклад 3.7: Знайти різницю двох двобайтних чисел без знаку. Операнди розташовуються в комірках третьої сторінки ЗПД з наступними адресами: перший операнд – 20*h*, 22*h*, другий операнд – 44*h*, 45*h*. Результат розмістити на місце зменшуваного.

```
; віднімання Z = W – Y
; W – вміст комірок ЗПД з адресами 20h, 22h (<20h>, <22h>)
; Y – вміст комірок ЗПД з адресами 44h, 45h(<44h>, <45h>)
SEL     RB0      ; вибір нульового банку регістрів
ANL     P2,#0    ; завантаження P2 := 00000000
ORL     P2,#8H   ;вибір третьої сторінки ЗПД
MOV     R0,#20H  ;завантаження покажчика адреси
MOV     R1,#22H  ;завантаження покажчика адреси
MOVX    A,@R0    ; A:=< 20h >
MOV     R5,A     ; R5 := W_CT
MOVX    A,@R1    ; A:=< 22h >
MOV     R4,A     ; R4 := W_МЛ
MOV     R1,#44H  ; завантаження покажчика адреси
MOV     R0,#45H  ;
MOVX    A,@R0    ; A:=< 45h >
CPL     A        ; A := Ā
CLR     C        ; C := 0
ADDC    A,#1     ; A := A+1+C (доповнювальний код)
MOV     R7,A     ;пересилка вмісту A в R7 (Y_МЛ)
MOVX    A,@R1    ; A:=< 44h >
CPL     A        ; A := Ā
ADDC    A,#0     ; A := A+C
MOV     R6,A     ; R6 := Y_CT
CLR     0        ; C := 0
MOV     A,R4     ; A := R4 – R7
ADDC    A,R7     ; R7 := Z_МЛ
MOV     R7,A     ;
```

```
MOV     A,R5      ; A := R5
ADDC    A,R6      ; A := R5 – R6
MOV     R6,A      ; R6 := Z_CT
```

***Приклад 3.8*:** Виконати маскування при введені інформації із порту *P*1. Переслати в регістр *R*5 інформацію з виводів порту *P*1[1], *P*1[2], *P*1[5], *P*1[6], *P*1[7]

```
IN      A,P1            ; введення байту із порту 1
ANL     A,#11100110b    ; маскування
MOV     R5,A            ; A := R5
```

***Приклад 3.9*:** Виконати логічний зсув вліво двобайтного слова, розміщеного в регістрах *R*5 і *R*6.

```
SEL     RBO       ; вибір нульового банку регістрів
CLR     C         ; C = 0
MOV     A,R6      ; A := R6
RLC     A         ; зсув молодшого байту
MOV     R6,A      ; пересилання результату
MOV     A,R5      ; A := R5
RLC     A         ; зсув старшого байту
MOV     R5,A      ; пересилка результату
```

***Приклад 3.10*:** Видати вміст акумулятора в послідовному коді через нульовий вивід порту *P*1[0], залишаючи без змін решту біт порту *P*1. Передачу вести, розпочинаючи з молодшого біту.

```
        MOV     R1,#8           ; лічильник бітів
LOOP:   JBO     LABEL 1         ; перехід, якщо біт A[0]
                                ; дорівнює одиниці
        ANL     P1,#11111110    ; скидання P1[0]
        JMP     LABEL 2         ;
LABEL1: ORL     P1,#1           ; встановлення P1[0]
        JMP     LABEL 2         ; надлишкова команда для
                                ; вирівнювання часу
```

```
                                    ; передачі 0 и 1
LABEL2: RR      A                   ; зсув A вправо (підготовка
        DYNZ    R1,LOOP             ; до передачі чергового біту)
```

3.5.3. Виконання команд передачі управління

Приклад 3.11: Передати управління за міткою *LABL*1, якщо перемикач банку регістрів (біт *PSW*[4]) дорівнює 1

```
Labl2:  MOV     A,PSW       ; пересилання PSW в акумулятор
        JB4     LABL1       ; перехід, якщо A[4]=1
        JMP     LABL2
LABL1:  MOV     A,R5        ; A:= R5
```

***Приклад 3.12*:** Обчислити значення функції $F = 2(R5 + R2)$. Якщо після виконання всіх дій ознака *C* дорівнюватиме одиниці, здійснити додавання одиниці до вмісту *R*7

```
        SEL     RB0         ; вибір нульового банку регістрів
        MOV     A,R2        ; A:= R2
        CLR     C           ; C = 0
        ADD     A,R5        ; A:= R2 + R5
        RLC     A           ; зсув вліво
        MOV     R5,A        ; R5:= A
        JC      DD1         ; якщо C=1, то перехід
        JMP     DD2         ;
DD1:    INC     R7          ; R7:= R7 +1
DD2:    NOP
```

***Приклад 3.13*:** Встановити вивід порту *P*2[7] в одиницю, за надходження на вхід *T*0 послідовності з восьми нульових імпульсів.

```
        MOV     P5,#8       ; завантаження в R5 числа
                            ; імпульсів
LABEL1: JT0     LABEL1      ; очікування сигналу 0 на вході T0
LABEL3  JT0     LABEL2      ; чекання сигналу 1 на вході T0
```

```
        JMP    LABEL3     ;
LABEL2  DYNZ   R5,LABEL1  ; повторювати доки не поступить
                          ; восьмий імпульс
        ORL    P2,#80h    ; встановлення одиниці на вході 7
                          ; порту P2
```

3.5.4. Розробка підпрограм виконання складних арифметичних операцій

Приклад 3.14: Розробити програму множення цілих восьмирозрядних чисел $Z = X \times Y$, де $(X,Y) < 1$. Множення реалізувати першим способом. У вихідному стані операнди подані в прямому коді.

Множення першим способом відбувається молодшими розрядами множника зі зсувом суми часткових добутків в сторону молодших розрядів за нерухомим множеним.

Операційна схема множення першим способом зображена на рис. 3.36, де *RG*1 – регістр множеного, *RG*2 – регістр множника, *RG*4 – регістр накопичування суми часткових добутків, *RG*3 – лічильник циклів. Алгоритм та цифрова діаграма стану регістрів зображені на рис.3.37 та рис. 3.38.

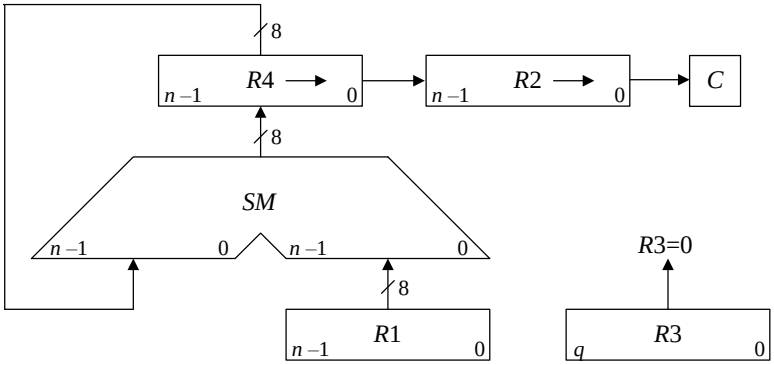


Рис. 3.36. Операційна схема множення чисел першим способом

Перед початком множення регістр *RG*4 встановлюється в нульовий стан. Під час множення в першому такті *i*-го циклу

аналізується значення молодшого розряду регістру множника *RG*2. Вміст *RG*1 додається до суми часткових добутків, що знаходяться в регістрі RG4, якщо *RG*2*(n)*=1, або не додається, якщо *RG*2*(n)*=0. В другому такті здійснюється правий зсув в регістрах *RG*1 і *RG*2. Після виконання *n* циклів молодші розряди 2*n*-розрядного добутку будуть записані в регістр *RG*2, а старші – у *RG*4.

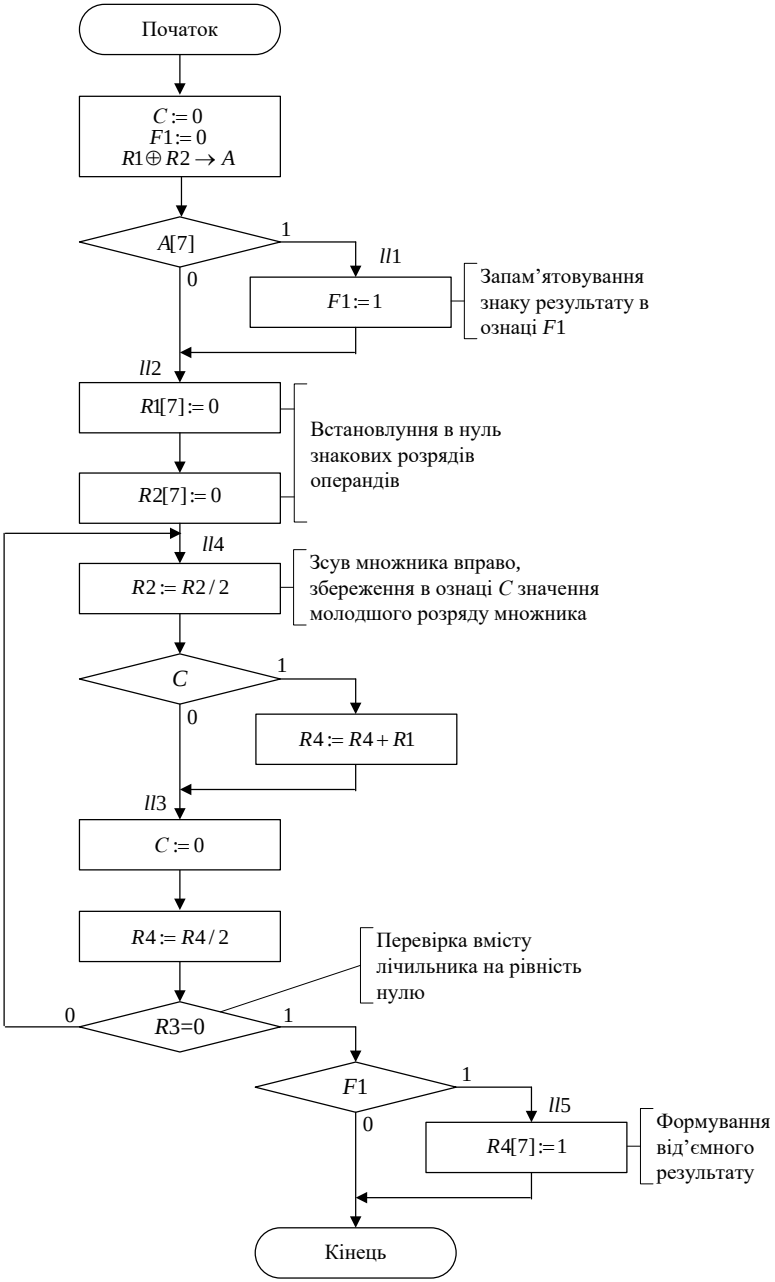


Рис. 3.37. Алгоритм виконання операції множення

	$\rightarrow R4$ [0 .. n]		$\rightarrow R2(X)$ [0 .. n]		$R1(Y)$ [0 .. n]		$R3$(Лічильник)
	0	0000	0	1011	0	1111	101
1	0	0000			0	1111	
	0	1111					
	0	1111					
	0	0111	1	0101			100
2	0	0111			0	1111	
	0	1111					
	1	0110					
	0	1011	0	1010			011
3					0	1111	
	0	0101	1	0101			010
4	0	0101			0	1111	
	0	1111					
	1	0100					
	0	1010	0	1010			001
5					0	1111	
	0	0101	0	0101			000

Рис.3.38. Цифрова діаграма виконання операції множення

```
; У вихідному стані операнд знаходиться в регістрі R1 та R2.
; Після виконання операції множення старші розряди добутку
; збережені в регістрі R4, молодші – в регістрі R2.
; R3 – лічильник циклів.
; Визначення знаку результату і подання операндів в ПК
      CLR   C          ; обнуління ознаки C
      CLR   F1         ; обнуління ознаки F1
      XCH   A,R1       ; обмін A та R1
      XRL   A,R2       ; ВИКЛЮЧНЕ АБО вмісту A та R2
      JB7   ll1        ; визначення знаку результату
      JMP   ll2
Ll1:  CPL   F1         ; встановлення ознаки F1
Ll2:  MOV   A,R2       ; A:=R2
      ANL   A,#7Fh     ; встановлення знакового розряду R2 в
                       ; нуль
      MOV   R2,A       ; R2:=A
```

```
        MOV   A,R1      ; A:=R1
        ANL   A,#7Fh    ; встановлення знакового розряду R1 в
                        ; нуль
        MOV   R1,A      ; R1:=A
; зсув множника вправо
Ll4:    MOV   A,R2      ; A:=R2
        CLR   C
        RRC   A         ; зсув
        MOV   R2,A      ; R2:=A
        JNC   Ll3       ; аналіз цифри множника, перехід якщо
                        ; A[0]=0
; додавання множника до суми часткових добутків
        MOV   A,R4
        ADD   A,R1      ; A:= R4 + R1
        MOV   R4,A      ; R4 := A
; зсув суми часткових добутків
Ll3:    CLR   C         ;C = 0
        MOV   A,R4      ; A:= R4
        RRC   A         ; зсув вправо, C := A[0]
        MOV   R4,A      ; R4:=A
; перевірка вмісту лічильника циклів, та
; повернення на початок циклу, якщо R3 ≠ 0
        DJNZ  R3,ll4    ; аналіз лічильника циклів
; перевірка знаку результату, якщо F1 = 1 встановлення
; в одиницю старшого розряду регістру результату
        JF1   Ll5                   ; аналіз знаку результату
        JMP   Ll6
Ll5:    MOV   A,R4                  ; A:=R4
        ORL   A,#80h                ; R4[7]:=1
        MOV   R4,A                  ; R4:=A
Ll6:    NOP
        END.
```

***Приклад 3.15*:** Розробити програму перетворення цілого восьмирозрядного двійкового числа в трирозрядне двійково-десяткове число.

```
; Перед початком перетворення двійкове число знаходиться в R3.
; Після перетворення код старшої десяткової цифри знаходиться в
; молодших розрядах R1, а коди двох молодших десяткових цифр
; відповідно в старшій і молодшій тетрадах регістру R2.
;
; початкові установки регістрів
        INC   R3
        MOV   R1,#0h
        CLR   A
        JMP   TST          ; перехід на команду перевірки
                           ; кінця циклу.
; цикл перетворення двійкового числа в двійково-десяткове
CYCLE:  ADD   A,#1h
        DA    A
        JNC   TST
        INC   R1
TST:    DJNZ  R3,CYCLE     ; декремент R3 і перевірка кінця
                           ; циклу.
        MOV   R2,A         ; пересилка молодших розрядів
                           ; результату в R2.
        END.
```

***Приклад 3.16*:** Розробити програму ділення правильних додатних двійкових дробів $Z=X/Y$, де $(X,Y)<1$, $X<Y$. Ділення виконати способом зі зсувом дільника.

Операційна схема та алгоритм ділення представлені на рис. 3.39 та рис. 3.40, відповідно. Цифрова діаграма виконання операції ділення зображена на рис. 3.41.

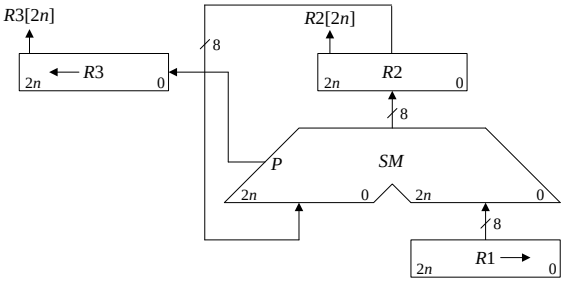


Рис.3.39. Операційна схема пристрою ділення із зсувом дільника

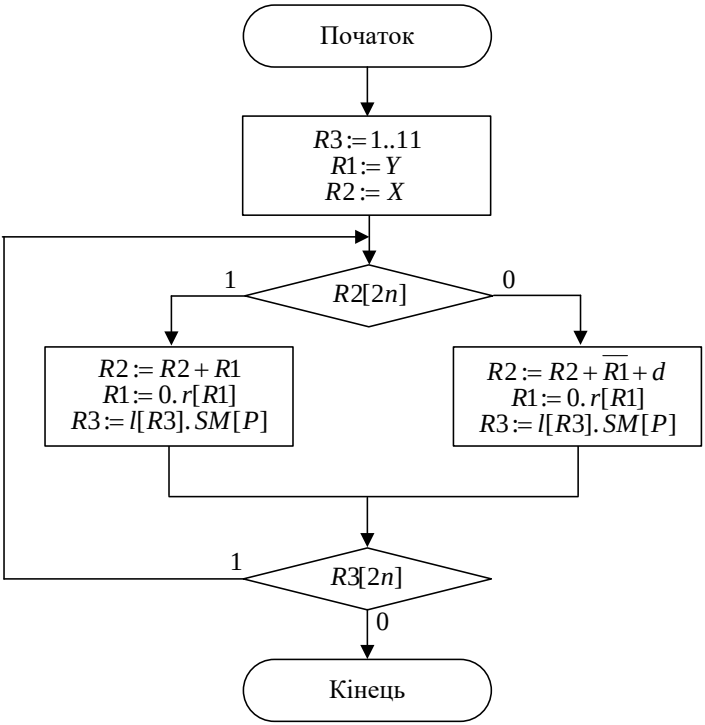


Рис. 3.40. Змістовний мікроалгоритм виконання операції ділення

№ такту	←R3 [2n .. 0]	R2(X) [2n .. 0]	R1(Y)→ [2n .. 0]	Логічна умова
	1 11111111	0 1001000	0 1100000	
1		0 1001000 1 0100000 1 1101000		R1[2n]=0
	1 11101111		0 0110000	R2[2n]=1
2	1 11101111	1 1101000 0 0110000 0 0011000	0 0110000	R1[2n]=1
	1 11011111		0 0011000	R2[2n]=1
3	1 11011111	0 0011000 1 1101000 0 0000000	0 0011000	R1[2n]=0
	1 10111111		0 0001100	R2[2n]=1

4	1	10111111	0	0000000	0	0001100	$R1[2n]=0$
			1	1110100			
			1	1110100			
	1	01101111			0	0000110	$R2[2n]=1$
5	1	01101111	1	1110100	0	0000110	$R1[2n]=1$
			0	0000110			
			1	1111010			
	0	11001111			0	0000011	$R2[2n]=0$

Рис.3.41. Цифрова діаграма виконання операції ділення

```
;У вихідному стані Y завантажений в регістрі R1, X – в регістрі R2,
; В регістрі R3 завантажуються маркерні одиниці.
; Результат формується в R3, під час зсуву його вмісту вліво
; чергова цифра результату заноситься у розряд, що звільнився.
; Операція ділення завершується, коли під час зсуву R3 в ознаку C
; запишеться нуль.
        SEL     RB0         ; вибір нульового банку регістрів
        MOV     R3,#FFh;    ; R3:=11111111
Ll5:    MOV     A,R2        ; A:=R2(X)
        JB7     Ll1         ; перевірка ознаки C (A[7])
        MOV     A,R1        ; пересилка R1 в A
        CLR     A           ; інвертування A
        ADD     A,#1        ; додавання одиниці до A
        ADD     A,R2        ; A:=(R2+(R1+1))
        MOV     R2,A        ; пересилка діленого
        JMP     Ll3         ;
Ll1:    MOV     A,R2        ; пересилка R2 в A
        ADD     A,R1        ; отримання суми діленого і
                            дільника
        MOV     R2,A        ; пересилка A в R2
Ll3:    MOV     A,R1        ; пересилка R1 в A
        CPL     C           ; встановлення в нуль ознаки C
        RRC     A           ; зсув дільника
        MOV     R1 A        ; пересилка A в R1
        CPL     C           ; встановлення в нуль ознаки C
        MOV     A,R2        ; інвертування старшого
                            ; розряду дільника (R2)
        JB7     Ll4
```

```
        CLR     C
Ll4:    MOV     A,R3        ; запис в молодший розряд
                            ; регістру R3
        RLC     A
        MOV     R3,A
        JB7     Ll5
                NOP
                END
```

Приклад 3.17: Розробити для МК48 програму обчислення квадратного кореня $A = \sqrt{B}$, де $0 \le B < 1$.

Найбільш простий алгоритм обчислення квадратного кореня з *n-розрядної* мантиси числа зводиться до підбору цифр результату розряд за розрядом, починаючи із старшого 2^{-1} розряду. При цьому обчислення *i*-ї цифри результату *X* відбувається таким чином. Після отримання чергової $(i-1)$-ї цифри в *i-й* розряд *A* розміщується одиниця. Обчислюється різниця $(B - A_i^2) = R_i$. Якщо, $R_i \ge 0$ то і A_i є число, де цифри всіх розрядів співпадають з цифрами результату *A*. Якщо, $R_i < 0$ то в *i*-му розряді необхідно поставити нуль і переходити до обчислення $(i+1)$ -го розряду. Оскільки в цьому випадку обчислення знову починається з підстановки пробної одиниці, то замість заміни одиниці на нуль в i-му розряді віднімається одиниця з $(i+1)$ -го.

Виконання обчислення з точністю до шостого розряду приведене на діаграмі (рис. 3.42, *а*). Для досягнення регулярності обчислень у разі отримання додатної різниці операцію віднімання доцільно замінити підсумуванням зворотного коду наступного результату з дописаними цифрами 11, при цьому отримаємо підсумовування в доповнювальному коді, наприклад, діаграма (рис. 3.42, *б*).

Для виконання обчислень в МК48 застосовуються восьмирозрядні регістри, тому корінь з восьмирозрядної мантиси можливо обчислити тільки з точністю до чотирьох розрядів після коми.

Операційна схема пристрою для обчислення квадратного кореня зображена на рис. 3.43, цифрова діаграма та алгоритм обчислення на рис. 3.44, та 3.45 відповідно.

```
0    00 10010001
   – 00 01
1    00 01010001
     00 10100010 зсув
   – 00 101
1    00 00000010
     00 00000100 зсув
   – 00 1101
0    11 00110100
     10 01101000 зсув
   + 00 11011
0    11 01000000
     10 10000000 зсув
   + 00 110011
0    11 01001100
     10 10011000 зсув
   + 00 1100011
0    11 01011110
```

$A = \sqrt{B} = 0{,}110000$

а

```
0    00 10010001
   + 11 11
1    00 01010001
     00 10100010 зсув
   + 11 011
1    00 00000010
     00 00000100 зсув
   + 11 0011
0    11 00110110
     10 01101100 зсув
   + 00 11011
0    11 01000100
     10 10001000 зсув
   + 00 110011
0    11 01010100
     10 10101000 зсув
   + 00 1100011
0    11 01011110
```

$A = \sqrt{B} = 0{,}110000$

б

Рис. 3.42. Діаграма обчислення квадратного кореня: *а* – у прямому коді; *б* – у доповнювальному коді.

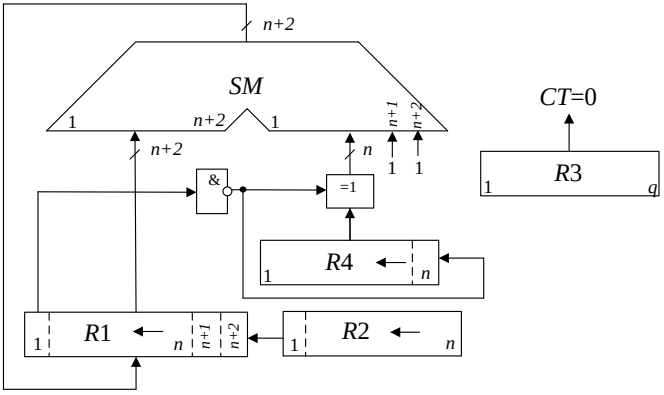


Рис. 3.43. Операційна схема обчислення квадратного кореня

№	← *RA*	← *RB*		← *RC*	*CT*	Мікрооперації
	[1 .. *n*]	[1 .. *n*]	*n*+1 *n*+2	[1 .. *n*]		
ПС	000000	000000	00	10010001	100	
1	000000	000000	10	01000100		2(*RGC* := *l*[*RGC*].0, *RGB* := *l*[*RGB*].*RGC*(0))
		+111111	11			*RGB* := *RGB* + $\overline{RGA}$.11
		000000	01			
2	00000,1	000000	10	10001000		*RGC* := *l*[*RGC*].0, *RGB* := *l*[*RGB*].$\underline{RGC}$(0), *RGA* := *l*[*RGA*].*RGB*(0)
		000001	01	00010000		*RGC* := *l*[*RGC*].0, *RGB* := *l*[*RGB*].*RGC*(0)
		+111110	11			*RGB*(0) = 0 ⇒ ____ *RGB* := *RGB* + $\overline{RGA}$.11
		000000	00		011	*CT* := *CT* −1; *CT* ≠ 0
3	0000,11	000000	00	00100000		*RGC* := *l*[*RGC*].0, *RGB* := *l*[*RGB*].$\underline{RGC}$(0), *RGA* := *l*[*RGA*].*RGB*(0)
		000000	00	01000000		*RGC* := *l*[*RGC*].0, *RGB* := *l*[*RGB*].*RGC*(0)
		+111100	11			*RGB*(0) = 0 ⇒ ____ *RGB* := *RGB* + $\overline{RGA}$.11
		111100	11		010	*CT* := *CT* −1; *CT* ≠ 0

4	000,110	111001	10	10000000		$RGC := l[RGC].0$, $RGB := l[RGB].\underline{RGC(0)}$, $RGA := l[RGA].RGB(0)$
		110011	01	00000000		$RGC := l[RGC].0$, $RGB := l[RGB].RGC(0)$
		+000110	11			$RGB(0) = 1 \Rightarrow$ $RGB := RGB + RGA.11$
		111010	00		001	$CT := CT - 1; CT \neq 0$
5	00,1100	110100	00	00000000		$RGC := l[RGC].0$, $RGB := l[RGB].\underline{RGC(0)}$, $RGA := l[RGA].RGB(0)$
		101000	00	00000000		$RGC := l[RGC].0$, $RGB := l[RGB].RGC(0)$
		+001100	11			$RGB(0) = 1 \Rightarrow$ $RGB := RGB + RGA.11$
		110100	11		000	$CT := CT - 1; CT = 0$

Рис.3.44. Цифрова діаграма обчислення квадратного кореня

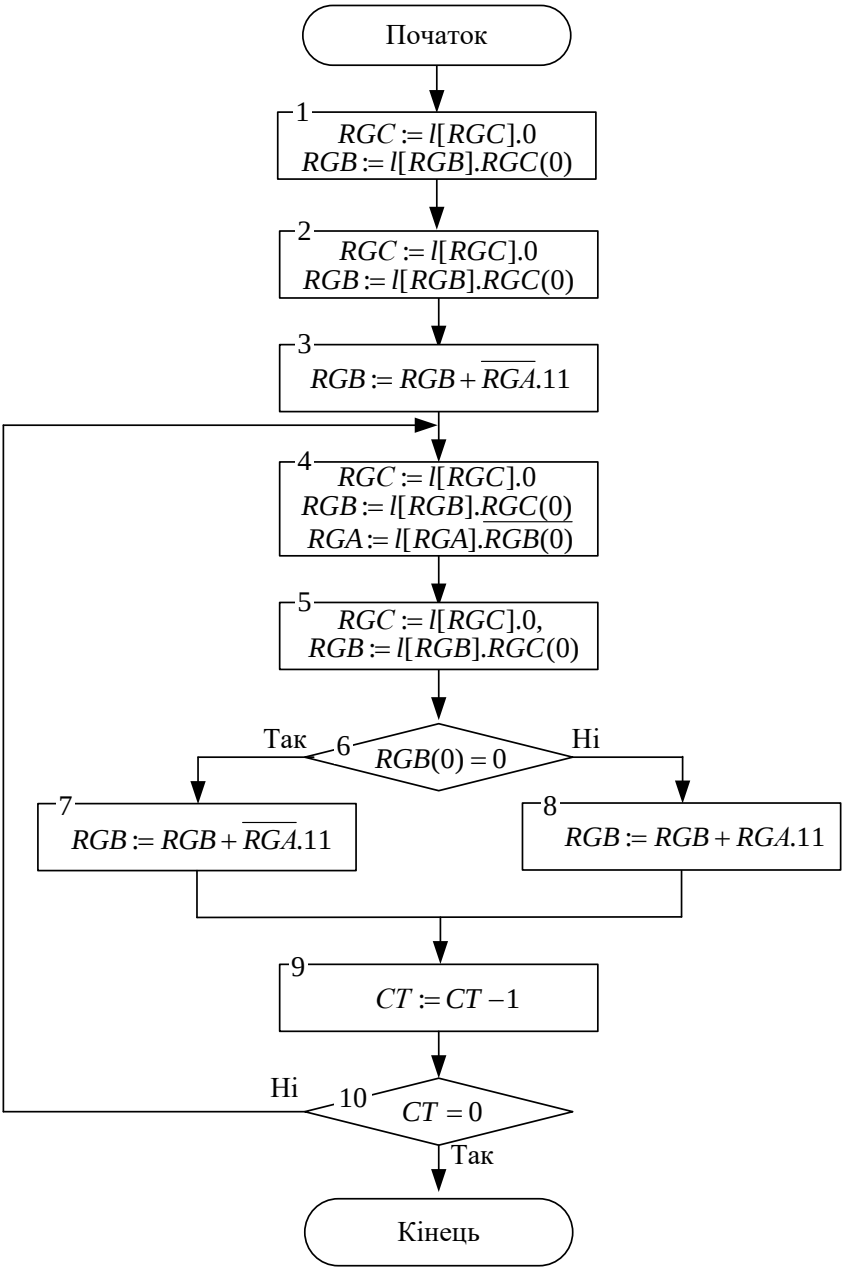


Рис. 3.45. Алгоритм обчислення квадратного кореня

```
; обчислення кореня з восьмирозрядної мантиси числа
; з точністю до чотирьох знаків після коми
      INS   A,BUS          ;
      MOV   R2,A           ; A – завантаження X
      MOV   R2,#10010001   ;
      MOV   R1,#0h         ; B – для підсумовування
      MOV   R0,#0h         ; C – для
                           ; зберігання результату
; завдання кількості повторень циклу CT:=4
      MOV   R3,#4h
LL3:  MOV   A,R0
      MOV   R4,A       ; R4:=A(R4:=A.11)
      MOV   R5,#0h     ; B[ZN]:=0
; два зсуви вліво A, B
      CLR   C
      MOV   A,R2
      RLC   A
      MOV   R2,A
      MOV   A,R1
      RLC   A
      MOV   R1,A
      CLR   C
      MOV   A,R2
      RLC   A
      MOV   R2,A
      MOV   A,R1
      RLC   A
      MOV   R1,A
      JB7   LL1        ; аналіз B[ZN]
      MOV   A,R4
      CPL   A
      MOV   R4,A       ; R4:=!A
LL1:  MOV   A,R4
      RLC   A
      RLC   A
      ORL   A,#3h
      MOV   R4,A       ; R4:=A.11
      ADD   A,R1
      MOV   R1,A
```

```
      JB7   LL2
      MOV   R5,#1h     ; !B[ZN]:=1
LL2:  CLR   C
      MOV   A,R0
      RLC   A
      ADD   A,R5       ; A[N]:=!B[ZN]
      MOV   R0, A
      DJNZ  R3,LL3     ; перевірка циклу
      MOV   A,R0
      OUTL  BUS,A
      END
```

3.5.5. Розробка програм управління

Приклад 3.18: Розробити програму реалізації алгоритму управління, що заданий логічною схемою алгоритм (ЛСА).

Вихідні дані:

- Алгоритм управління:

$$\text{Н}\ Y_5 X_1 \overset{1}{\uparrow}(Y_1 Y_2) \overset{2}{\uparrow} \overset{1}{\downarrow}(Y_3 Y_4) \overset{2}{\downarrow}\ \text{К}$$

- Тривалість управляючих сигналів:

$$T(y_1) = T(y_2) \geq 240\,мкс\ ,\ \text{T}(\text{y}_3) = T(y_4) \geq 30\,мкс\ ,\ \text{T}(\text{y}_5) = 450\,мкс$$

Алгоритм управління зображений на рис. 3.46.

Для вводу і виводу сигналів будемо використовувати порт *P*1, причому, розряди порту *P*1[6] та *P*1[7] в початковому стані налаштовані на ввід ($P1[6,7]=1$), а *P*1[5..0] – на вивід інформації. Відповідність виходів порту і сигналів вказано в табл. 3.6.

Таблиця 3.6. Відповідність виходів порту і сигналів

Розряд порту	*P*17	*P*16	*P*15	*P*14	*P*13	*P*12	*P*11	*P*10
Сигнал	*X*1	Вхід	*Y*1	*Y*2	*Y*3	*Y*4	*Y*5	Вихід

Для формування потрібної тривалості сигналів *Y*1 і *Y*2 будемо використовувати *TCNT* в режимі таймера. При *F*=6 *МГц* для відліку проміжку часу, не меншого 240 мкс, необхідна зміна змістовного *TCNT* на 3. Для формування інтервалу часу, тривалістю 30 мкс

(керуючі сигнали *Y*3 та *Y*4), при вказаній частоті *F* потрібно виконати 12 командних циклів. Для формування затримки в 450 мкс (*Y*5), необхідно використовувати і таймер, і тривалість команди.

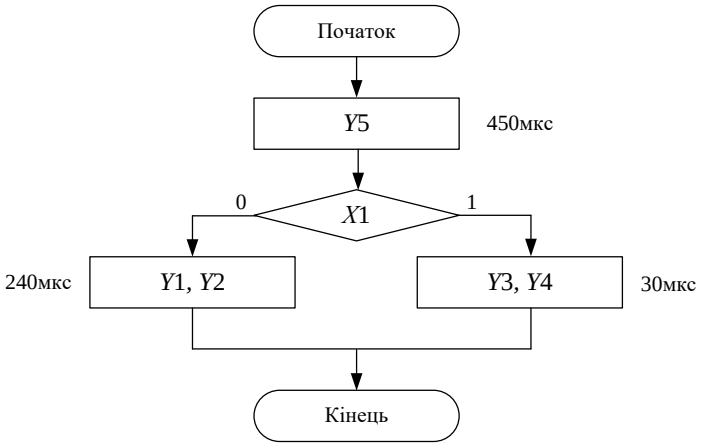


Рис. 3.46 Алгоритм управління

```
;Перед виконанням програми порт P1 знаходиться в стані 11000000.
;Формування часового інтервалу - 450 мкс
          MOV   R5,#10      ; завантаження константи 10 в ;R5
          MOV   A,#FBH      ; завантаження константи
                            ; (– 5)ДК в таймер
          MOV   T,A
          ORL   P1,#2H      ; встановлення сигналу Y5
          STRT  T           ; запуск таймера
Label2:   JTF   Label1      ; відлік часового інтервалу
          JMP   Label2      ; з використанням таймеру
                            ; (400мкс)
Label1:   DJNZ  R5,label1   ; відлік часового інтервалу
                            ; з використанням R5(50мкс)
          ANL   P1,#C0h     ; зняття управляючого сигналу
          IN    A,P1        ; ввід та перевірка X1
          JB7   Label3
          MOV   A,#FDh      ; завантаження константи (– 3)ДК
                            ; в таймер
          MOV   T,A
          ORL   P1,#30h     ; встановлення сигналів
                            ; Y1 = Y2 = 1
```

```
        STRT  T           ; запуск таймера
Label4: JTF   Label5      ; відлік часових інтервалів з
        JMP   Label4      ;використанням таймеру ;(240мкс)
Label5: ANL   P1,#C0h     ; зняття керуючих сигналів
                          ; Y1 = Y2= 1
        JMP   Label6
Label3: MOV   R5,#4       ; завантаження константи 4 в R5
        ORL   P1,#0Ch     ; встановлення сигналів
                          ; Y3 = Y4 = 1
label4: OP                ; відлік часового інтервалу
        DJNZ  R5,label4   ; з використанням R5 (30мкс)
        ANL   P1,#C0h     ; зняття управляючих сигналів
Label6: NOP
        END.
```

***Приклад 3.19*:** Розробити структурну схему підключення до МК48 програмованого периферійного адаптера ВВ55. Розробити програму пересилки даних із порту *PB* в порти *PA* та *PC*. Адреси портів ППА належать загальному адресному простору зовнішньої пам'яті даних.

Вихідні дані:

- Адреси портів:

  PA – ACh, PB – ADh, PC – AEh, Регістр УСРР – *AFh.*

Структурна схема підключення до МК48 однієї сторінки ЗПД та програмованого периферійного адаптера ВВ55 зображена на рис. 3.47. Для підключення ППА застосовується селектор адреси СА.

; Прийом байту із порту *PB* в порти *PA* та *PC*.

```
MOV     R0,#0AFh      ; адресу Регістру УСРР завантажуємо у
                      ; покажчик адреси
MOVX    A,#082h       ; ініціалізація ВВ55 (порти PA і PC
                      ; налаштовуються на вивід, а порт PB –
                      ; на ввід даних)
MOV     @R0,A         ; завантаження управляючого слова в
                      ; регістр УСРР
```

```
MOV     R0,#0ADh      ; підготовка адреси порту PB
MOVX    A, @R0        ; читання даних із порту PB
MOV     R0,#0ACh      ; підготовка адреси порту PA
MOVX    @R0,A         ; запис даних в порт PA
MOV     R1,#0AEh      ; підготовка адреси порту PC
MOVX    @R1,A         ; запис даних в порт PC
END
```

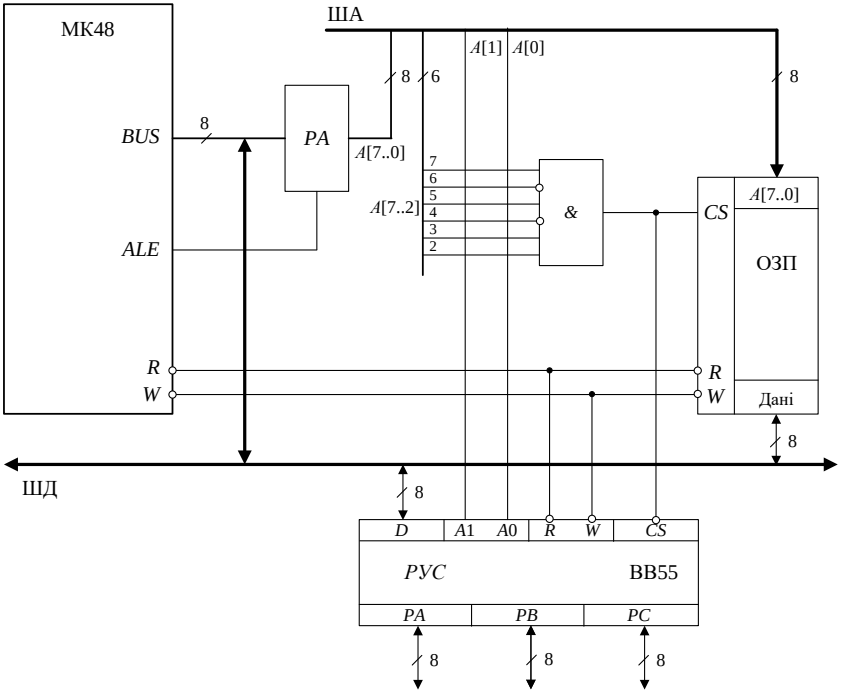


Рис. 3.47. Структурна схема підключення програмованого периферійного адаптера до МК48

***Приклад 3.20*:** Розробити структурну схему підключення до МК48 ЗПД та програмованого зв'язкового адаптера ВВ51. Зовнішня пам'ять даних складається з чотирьох сторінок, для перемикання між якими застосовуються два молодших виводи порту *P*1. Адреси регістрів УСРР та УСК ПЗА належать першій сторінці ЗПД (адреса РД – 00*h*; адреса РС(УС) – 01*h*)

Розроблена структурна схема МПС зображена на рис. 3.48.

```
; P1 = 11111111 встановлюється під час ініціалізації
          ANL    P1,#0FDh ; вибір номера сторінки
          ORL    P1,#01h  ; ”1” 11111101 для ВВ51
          SEL    RB1      ; установка першого банку
                          ; регістрів
          MOV    R0,#00h  ; адреса регістра даних ПЗА
                          ; завантажується в R0
          MOV    R1,#01h  ; адреса регістру УСРР (УСРР,
                          ;УСІ) ПЗА завантажується в R1
          MOV    A,#07Eh  ; ініціалізація ПЗА
                          ; (асинхронний режим).
          MOVX   @R1,A    ; передача байта в послідовний
                          ; канал з регістра R2
          MOV    A;#31h   ; запис УСК (УСІ) в PG ПЗА
          MOVX   @R1,A
WW1:      MOVX   A,@R1    ; читання слова стану ПЗА
                          ; і перевірка готовності передавача
          ANL    A,#01
          JZ     WW1
          MOV    A,R2     ; запис даних в передавач ПЗА
                          ; з R2
          MOVX   @R0,A    ; прийом байта з послідовного
                          ; каналу в R2
          MOV    A,#34h   ; запис управляючого слова в ПЗА
          MOVX   @R1,A
WW2:      MOV    X,a,@R1  ; читання слова стану з ПЗА
                          ; і перевірка готовності приймача
          ANL    a,#02h
          JZ     WW2
          MOVX   A,@R1    ; читання стану слова в ПЗА
                          ; і перехід, якщо знайдена помилка
          ANL    A,#38h
          JNZ    ERROR
          MOVX   A,@R0    ; читання байта даних та передача
                          ; вR2
          MOV    R2,A     ;
ERROR:    NOP             ; обробка помилки
          END
```

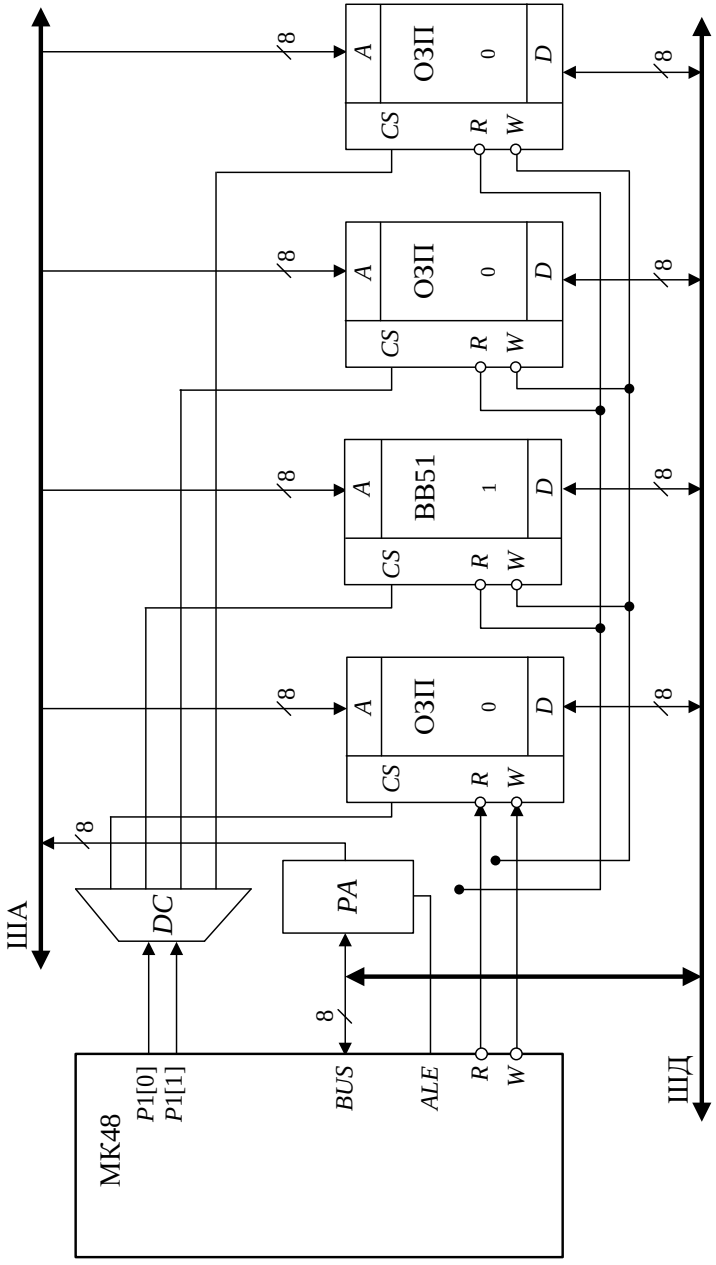


Рис. 3.48. Структурна схема підключення програмованого зв'язкового адаптера до МК48

4. ОДНОКРИСТАЛЬНИЙ МІКРОКОНТРОЛЕР КР1816ВЕ51

4.1. Загальна характеристика

До сімейства однокристальних мікроконтролерів належать мікросхеми серії КР1816 ВЕ51, ВЕ31, ВЕ751, а також мікросхеми серії КР1830 ВЕ31 і ВЕ51. Організація і система команд мікросхем зазначених серій ідентична. Відмінність полягає в організації внутрішньої пам'яті програм і величині струму живлення (табл. 4.1). Як базову з сімейства однокристальних мікроконтролерів розглянемо мікросхему КР1816ВЕ51.

Таблиця 4.1. Параметри мікроконтролерів сімейства КМ1816

МК	**Об'єм РПП, байт**	**Тип РПП**	**Струм живлення, *mA***
КР1816ВЕ31	–	–	150
КМ1816ВЕ51	4К	ПЗУ	150
КМ1816ВЕ751	4К	ППЗУ	220
КР1830ВЕ31	–	–	18
КР1830ВЕ51	4К	ПЗУ	18

4.2. Архітектура мікроконтролера КР1816ВЕ51

4.2.1. Умовне графічне позначення мікросхеми

Умовне позначення мікросхеми КР1816ВЕ51 показане на рис. 4.1. Мікроконтролер конструктивно розміщений в корпусі з сорока зовнішніми виходами. Нижче приводяться символічні імена виходів мікросхеми.

GND – загальний;

Vcc – напруга живлення +5В;

*BQ*1 – вхід для підключення зовнішнього джерела синхронізації (для серії КР1830) або кварцового резонатора;

*BQ*2 – вхід для підключення зовнішнього джерела синхронізації (для серії КР1816) або кварцового резонатора;

RST	–	вхід сигналу загального скидання;
EMA	–	вхід дозволу зовнішній пам'яті; вхід для подачі імпульсів під час програмування ППЗУ;
PME	–	дозвіл читання зовнішньої пам'яті програм;
ALE	–	строб адреси зовнішньої пам'яті;
*P*0	–	восьмирозрядний двоспрямований порт вводу-виводу;
*P*1, *P*2, *P*3	–	восьмирозрядні квазідвоспрямовані порти вводу-виводу;

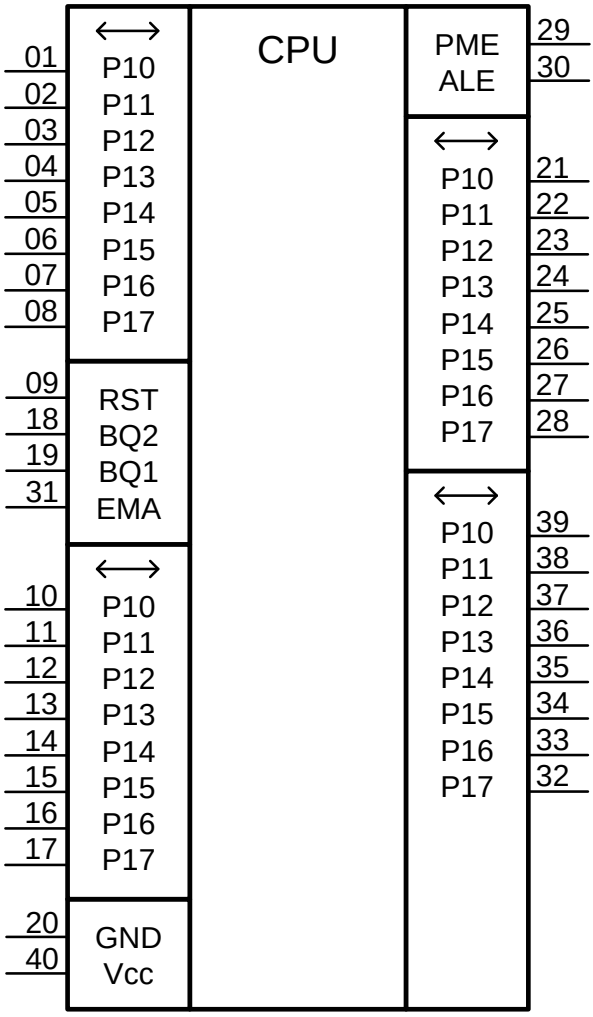


Рис. 4.1. Умовне графічне позначення мікроконтролера КР1816ВЕ51

Кожна лінія порту *P*3 має альтернативну функцію:

*P*30 – *R*×*D*	–	вхід послідовного порту;
*P*31 – *T*×*D*	–	вихід послідовного порту;
*P*32 – *INT*0	–	вхід 0 сигналу запиту переривання від зовнішнього джерела;
*P*33 – *INT*1	–	вхід 1 сигналу запиту переривання від зовнішнього джерела;
*P*34 – *T*0	–	вхід лічильника зовнішніх подій *Т/Л*0;
*P*35 – *T*1	–	вхід лічильника зовнішніх подій *Т/Л*1;
*P*36 – *WR*	–	вихід сигналу запису в зовнішню пам'ять даних;
*P*37 – *RD*	–	вихід сигналу читання із зовнішньої пам'яті даних;

За електричними параметрами всі сигнали мікросхеми КР1816ВЕ51 сумісні з сигналами інтегральних мікросхем, виготовлених за технологією ТТЛ – транзисторно-транзисторна логіка.

4.2.2. Структура мікроконтролера

Узагальнена структурна схема МК51 показана на рис. 4.2.

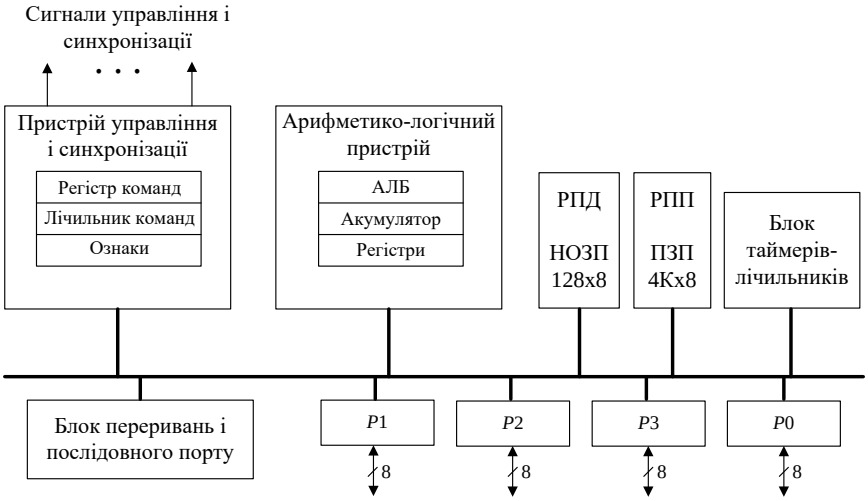


Рис.4.2. Структурна схема мікроконтролера МК51

Мікроконтролер містить резидентну пам'ять програм (РПП) та резидентну пам'ять даних (РПД); пристрій управління і синхронізації, до складу якого входить лічильник команд, регістр команд і регістр ознак; арифметико-логічний пристрій, до складу якого входить АЛБ, акумулятор і регістри; блок таймерів-лічильників та блок послідовного інтерфейсу і переривань. Обмін даними здійснюється через чотири порти *P*0, *P*1, *P*2, *P*3, або через послідовний порт.

Резидентна пам'ять програм

Резидентна пам'ять програм, має ємність 4Кб. Призначена для зберігання команд, констант, управляючих слів ініціалізації, таблиць кодування вхідних і вихідних змінних. Резидентна пам'ять даних підключена до шістнадцятибітної шини адреси, що надходить з лічильника команд, або регістру покажчика даних.

Резидентна пам'ять даних

Резидентна пам'ять даних призначена для зберігання змінних у процесі виконання програми, адресується одним байтом і має ємність 128 байт. До адресного простору резидентної пам'яті даних належать регістри спеціальних функцій.

Арифметико-логічний пристрій

Восьмибітний АЛП виконує арифметичні операції додавання, віднімання, множення, ділення, логічні операції – І, АБО та ВИКЛЮЧНЕ АБО, операції циклічного зсуву, скидання, інвертування, інкременту, декременту, тощо.

Важливою особливістю АЛП МК51 є можливість оперувати не тільки байтами, але і бітами. Окремі програмно доступні біти можуть бути встановлені, скинуті, передані, інвертовані, проаналізовані, використані в логічних операціях.

Під час виконання багатьох команд а АЛП формуються ряд ознак, які фіксуються у регістрі слова стану програми, що належить до регістрів спеціальних функції. Ознака переносу С приймає участь і модифікується в процесі виконання великої кількості операції в АЛП, таких як додавання, віднімання, зсув, тощо. Окрім того ознака переносу виконує дії своєрідного акумулятора під час виконання операцій з бітами. Ознака переповнення *OV* фіксує автоматичне переповнення під час виконання операцій зі знаками, і дає можливість реалізації арифметики в доповнювальних кодах.

Таким чином АЛП може оперувати з чотирма типами інформаційних об'єктів: булевськими (1 біт), цифровими (4 біти), байтовими (8 біт), і адресними (16 біт). В АЛП виконується 51 операція пересилки та перетворювання даних. Застосовуються одинадцять режимів адресації – сім для даних, чотири для адресів. Шляхом комбінування операцій з різними режимами адресації базова кількість команд 111 розширюється до 255 із 256 можливих за застосування однобайтного коду операції.

Регістри спеціальних функцій

Модель програміста МК51 зображена на рис. 4.3. На моделі програміста зображені умовні позначення регістрів спеціальних функцій (*SFR*):

DPTR – регістр-покажчик даних;
*ACC** – акумулятор,
*B** – регістр-розширювач акумулятора
SP – покажчик стека
*PSW** – слово стану програми
*P*0* – порт 0
*P*1* – порт 1
*P*2* – порт 2
*P*3* – порт 3
*IP** – регістр пріоритетів
*IE** – регістр маски переривань
TMOD – регістр режиму таймерів/лічильників
*TCON** – регістр управління/статусу таймера
*TH*0 – таймер 0 (старший байт)
*TL*0 – таймер 0 (молодший байт)
*TH*1 – таймер 1 (старший байт)
*TL*1 – таймер 1 (молодший байт)
*SCON** – регістр управління приємопередавачем
SBUF – буфер приємопередавача;
PCON – регістр управління потужністю.

Одинадцять регістрів спеціальних функцій допускають як байтову, так і побітову адресацію. Адреси регістрів *SFR* приведені

на рис. 4.3, регістри, помічені позначкою (*), допускають адресацію окремих біт.

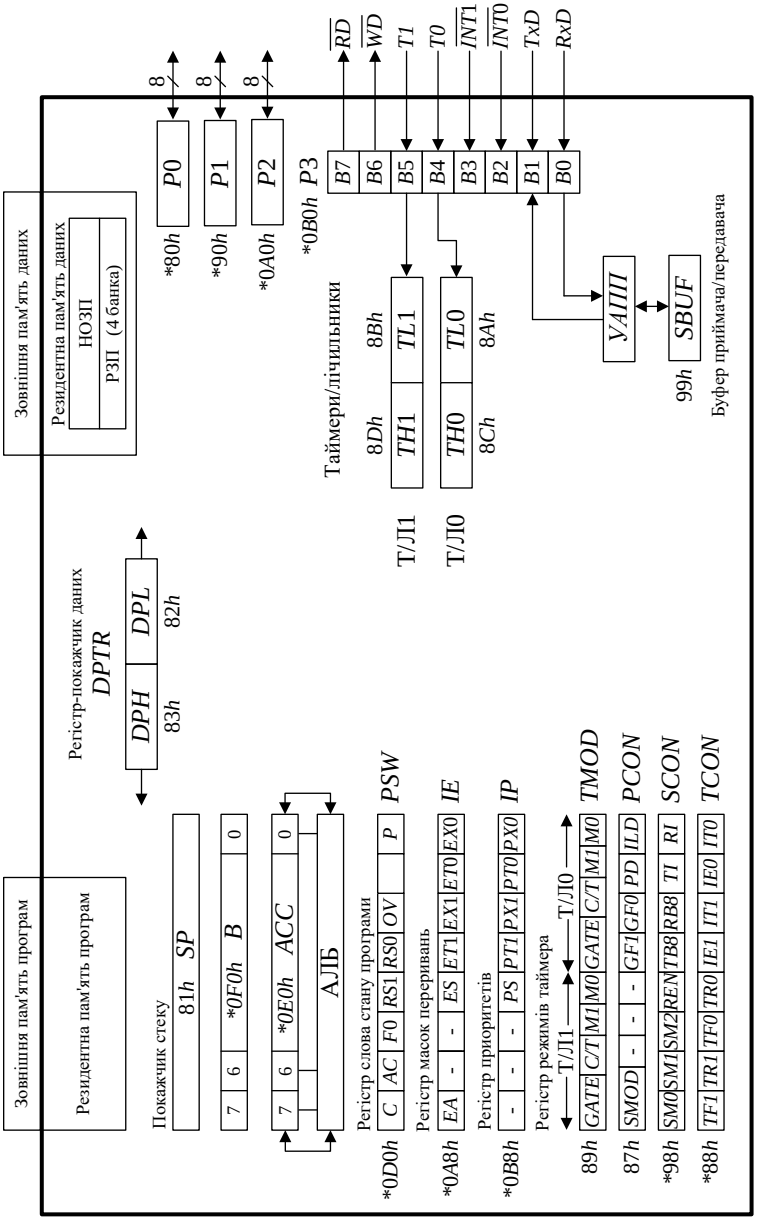


Рис.4.3. Модель програміста

Акумулятор *ACC* це восьмирозрядний регістр, який використовується як приймач або джерело операнда під час виконання арифметичних і логічних операцій та ряду операцій передачі даних. За застосування акумулятора виконуються операції зсувів, перевірки на нуль, формування ознаки парності (паритету).

Регістр B

Регістр *B* використовується під час виконання операцій множення і ділення. Під час виконання інших операцій може застосовуватись як допоміжний регістр.

Регістр слова стану програми PSW

Регістр слова стану програми *PSW* призначений для зберігання інформації про ознаки, що формуються в АЛП під час виконання обчислень. Наведемо перелік розрядів регістра (див. модель програміста на рис. 4.3):

C (*PSW*.7) – ознака переносу, встановлюється і скидається апаратно або програмно при виконанні арифметичних та логічних операцій, якщо в старшому розряді результату виникає перенос або позика;

AC (*PSW*.6) – ознака додаткового переносу встановлюється і скидається апаратно при виконанні додавання і віднімання, якщо в третьому біті результату виникає перенос або позика.

F0 (*PSW*.5) – ознака нуля, встановлюється і скидається програмно, специфікується користувачем;

*RS*1 (*PSW*.4) – вибір банку регістрів, встановлюється і скидається
*RS*2 (*PSW*.3) програмно для вибору банку регістрів (рис. 4.3);

OV (*PSW*.2) – ознака переповнювання, встановлюється і скидається апаратно за виникнення переповнення під час виконання арифметичних операцій з цілими числами зі знаком;

(*PSW*.1) – не використовується;

P (*PSW*.0) – ознака парності, встановлюється і скидається апаратно в кожному циклі команди і фіксує парну/непарну кількість одиниць в акумуляторі, тобто біт виконує контроль за парністю.

Всі біти окрім біта *P* доступні програмно для запису або читання, біта *P* доступний тільки для читання.

Покажчик стека

Восьмибітний покажчик стека *SP* може адресувати будь-яку область РПД і призначений для зберігання адреси комірки стека, до якого було останнє звернення. В мікроконтролері реалізований передінкрементний/постдекрементний спосіб адресації стеку. Вміст регістру *SP* інкрементується перш ніж дані будуть запам'ятовуватись у стеку під час виконання команд `PUSH` і `CALL`, та декрементується після виконання команд `POP` і `RET`.

В процесі ініціалізації мікроконтролера в регістр *SP* автоматично завантажується код 07*h*. Якщо прикладна програма не перевизначає стек, то перший елемент даних у стеку буде розміщуватись у РПД за адресою 08*h*.

Регістр-покажчик даних

Двобайтний регістр-покажчик даних *DPTR* застосовується для фіксації шістнадцятибітної адреси під час виконання операцій звернення до зовнішньої пам'яті. Командами мікроконтролера регістр-покажчик даних може бути застосований і як шістнадцятибітний регістр і як два незалежних восьмибітних регістра.

Таймер/лічильник

У склад мікроконтролера входять регістрові пари *TH*0, *TL*0, *TH*1, *TL*1, на основі яких функціонують два незалежних шістнадцятибітних таймера/лічильника подій.

Буфер приймача/передавача

Буфер приймача/передавача *SBUF* складається з двох незалежних регістрів – буферу приймача і буферу передатчика. Завантаження байта в регістр викликає початок процесу передачі через послідовний порт. Коли байт зчитується із регістру, його джерелом є приймач послідовного порту.

Регістри спеціальних функції *IP*, *IE*, *TMOD*, *TCON*, *SCON*, *PCON* застосовуються для фіксації і програмної зміни управляючих бітів, бітів стану схеми переривання, таймера/лічильника, приймача/передавача послідовного порту, а також для управління потужністю електроживлення мікроконтролера.

Пристрій управління та синхронізації

Пристрій управління та синхронізації є кварцовим резонатором, що підключається до зовнішніх виводів *X*1 та *X*2 корпуса мікросхеми МК51 управляє роботою внутрішнього генератора, який в свою чергу формує сигнали синхронізації.

Пристрій управління на основі сигналів синхронізації формує машинний цикл фіксованої довжини, що дорівнює дванадцяти періодам резонатора і відповідає шістьом станам управляючого автомата (*S*1 – *S*6). Кожний стан управляючого автомата вміщує дві фази *P*1, *P*2 сигналів резонатора. В фазі *P*1, зазвичай, виконуються операції в АЛП, а в фазі *P*2 здійснюється міжрегістрова передача. Весь машинний цикл складається з 12 фаз, розпочинаючи з фази *S*1*P*1 і закінчуючи фазою *S*6*P*2 (рис. 4.4).

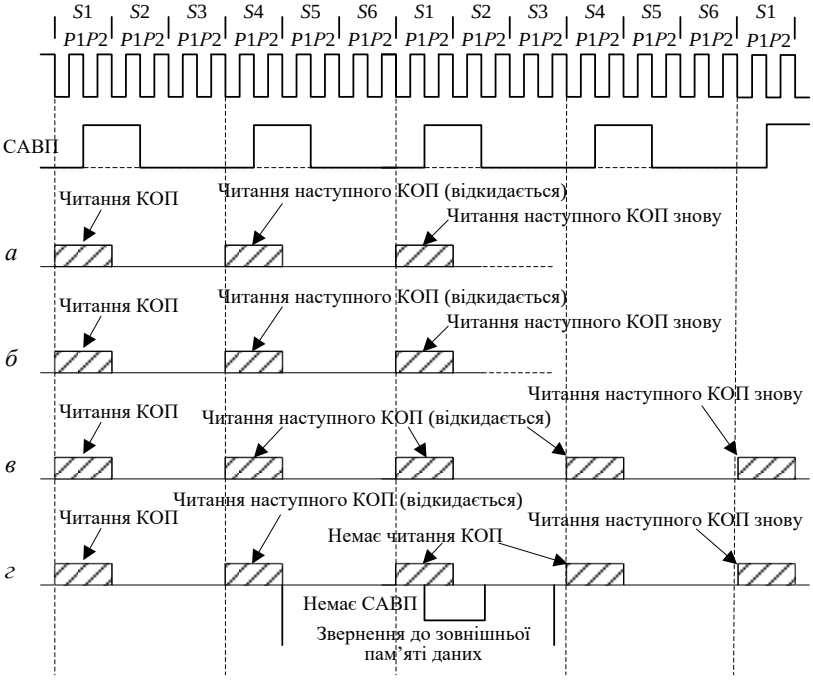


Рис. 4.4. Послідовність вибірки та виконання команд в МК51: *а* – команда 1 байт/1 цикл (наприклад, *INC A*); *б* – команда 2 байти/1 цикл (наприклад,

ADD A,#d); *в* – команда 1 байт/2 циклу (наприклад, *INC DPTR*); *г* – команда 1 байт/2 циклу (*MOV X*)

Часова діаграма ілюструє роботу пристрою управління під час вибірки та виконання команд різної складності. Всі заштриховані сигнали є внутрішніми і недоступними для зовнішнього контролю. Видимими є тільки сигнали резонатора і сигнали стробу адреси зовнішньої пам'яті (САЗП). Сигнали САЗП формуються два рази за один машинний цикл і застосовуються для управління процесом звернення до зовнішньої пам'яті.

Більшість команд мікропроцесора МК51 виконується за один машинний цикл. Команди, що оперують з двобайтними словами або пов'язані зі зверненням до зовнішньої пам'яті виконуються за два машинні цикли. Команди ділення та множення потребують чотири машинні цикли.

4.2.3. Організація пам'яті

Мікроконтролер МК51 має фізично і логічно розділену пам'ять програм і пам'ять даних, ємність кожної з котрих може бути розширена до 64 Кб за рахунок підключення зовнішніх мікросхем пам'яті. Пам'ять програм і пам'ять даних мають різні механізми адресації і працюють під управлінням різних управляючих сигналів. Організація пам'яті ілюструється на рис. 4.5.

Звернення до пам'яті програм відбувається за управляючим сигналом *PME*, до пам'яті даних – за сигналами *WR, RD*.

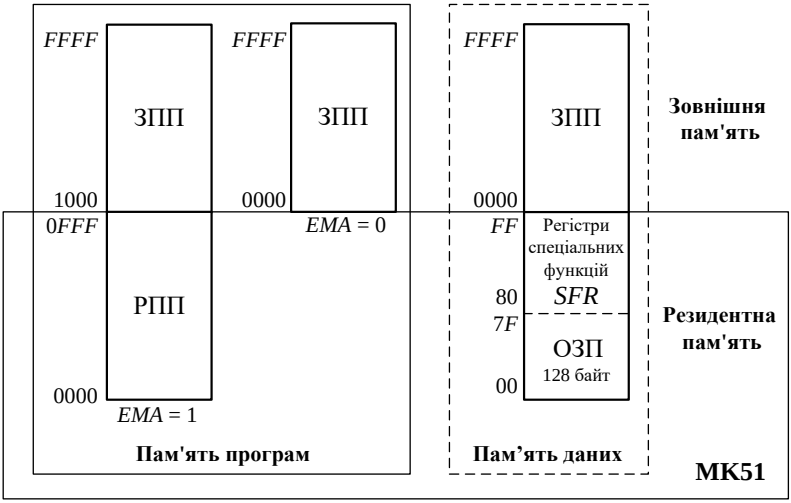


Рис 4.5. Організація пам’яті в архітектурі МК51

Пам’ять програм

Пам'ять програм поділяється на *резидентну* (*внутрішню*) *пам’ять програм* (РПП), що знаходиться всередині мікросхеми і *зовнішню пам’ять програм* (ЗПП), для реалізації якої потрібні додаткові мікросхеми пам'яті. Резидентна пам’ять програм може бути відключена за встановлення низького рівня сигналу на вході *EMA* = 0. Організація резидентної пам’яті програм за встановлення сигналу *EMA* = 1 зображена на рис. 4.6. Адреси 0000*h*, 0003*h*, 00*Bh*, 0013*h*, 001*Bh* і 0023*h* мають спеціальне призначення. З адреси 0000*h* розпочинає виконуватися програма під час системного скидання. Інші адреси призначені для зберігання початкових адрес підпрограм обслуговування переривань від зовнішніх сигналів, таймерів/лічильників або послідовного інтерфейсу.

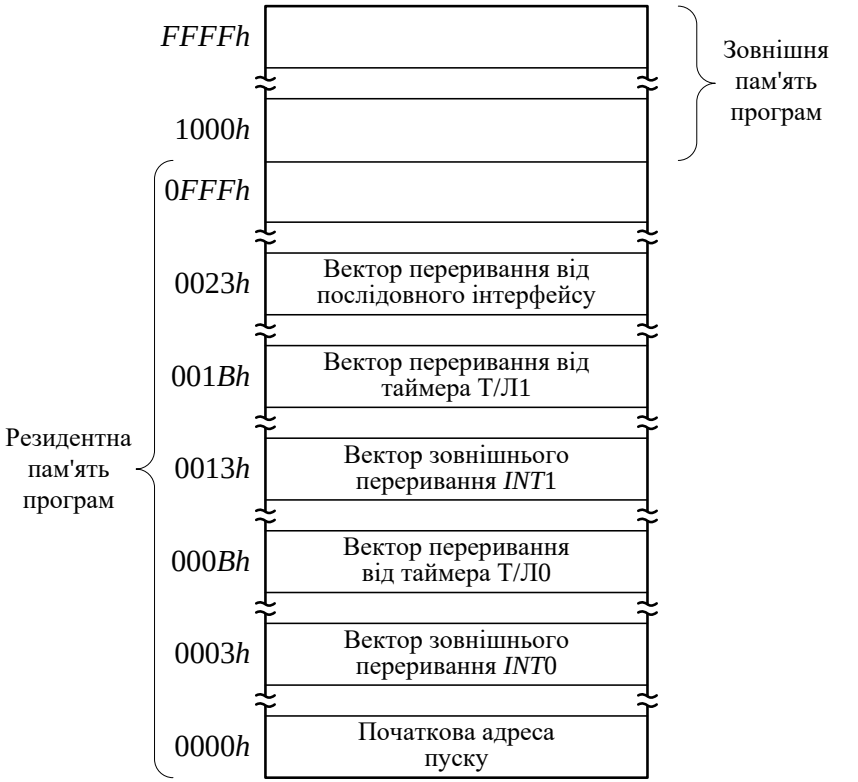


Рис. 4.6. Організація резидентної пам’яті програм

Під час звернення до зовнішньої пам'яті програм старший байт адреси передається через порт *P*2. Видача молодшого байта адреси, а також і передавання байта даних здійснюється через порт *P*0 в режимі часового мультиплексування.

Для адресації операндів в пам'яті програм застосовується безпосередня адресація або непряма базова індексна адресація. За безпосередньої адресації з порту *P*0 пам'яті програм вибирається константа, явно задана в команді. Наприклад, під час виконання команди `MOV R2,#15` в регістр пересилається константа 15 (0*Fh*).

За непрямої базової індексної адресації операндів в якості індексного регістру використовується акумулятор, а в якості базового – регістр-покажчик даних *DPTR* або лічильник команд *PC*. Пересилання операндів виконується за допомогою команди `MOVC`.

Підключення зовнішньої пам’яті програм для випадку, коли *EMA* = 0, показано на рис. 4.7, де *D* та *A* входи даних та адреси відповідно.

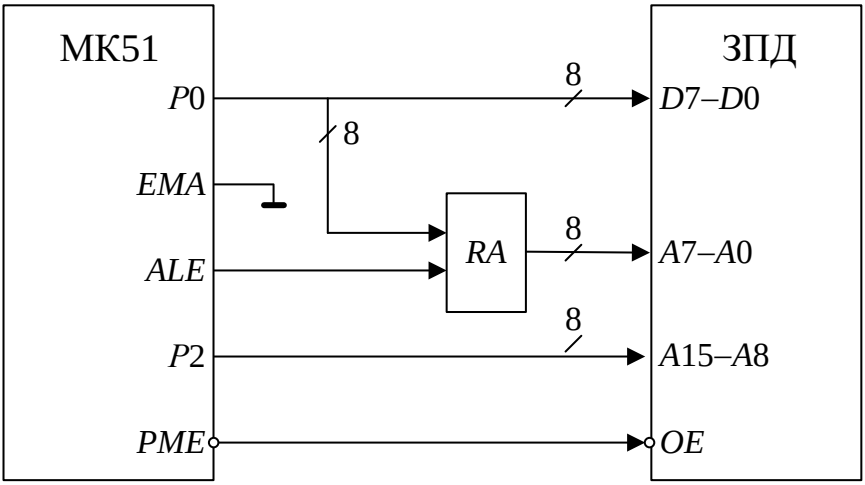


Рис 4.7. Схема підключення зовнішньої пам'яті програм

Пам’ять даних

Пам'ять даних як і пам'ять програм поділяється на *резидентну* (*внутрішню*) *пам’ять даних* (РПД) і *зовнішню пам’ять даних* (ЗПД). Резидентна пам'ять *даних* складається з двох областей – оперативного запам’ятовуючого пристрою ОЗП об’ємом 128 байт з адресами 00*h* – 7*Fh* і регістрів спеціальних функцій *SFR* з адресами 80*h*–*FFh* (рис. 4.6).

Оперативний запам’ятовуючий пристрій містить чотири банки регістрів загального призначення (рис. 4.8). Вибір банків регістрів 0 – 3 визначається значенням двох бітів *RS*1 та *RS*0 регістру *PSW* (рис. 4.3).

Команди програми можуть звертатися до регістрів будь-якого банку за іменами регістрів *R*7 – *R*0, тобто з використанням прямої регістрової адресації. Це дозволяє економити пам'ять програм, оскільки команди, у даному випадку мають меншу довжину ніж при застосуванні непрямої адресації.

Шістнадцять байт ОЗП, з адресами 20*h* – 2*Fh* утворюють область комірок, до яких можливе застосування побітової адресації (рис. 4.8). До комірок цієї області зручно звертатися за допомогою

спеціальних команд роботи з бітами, використовуючи при цьому пряму адресацію.

Доступ до байтів ОЗП можливий за допомогою непрямої регістрової або прямої адресації. У першому випадку як покажчики використовуються регістри *R*0, *R*1 вибраного банку регістрів. Наступна команда ілюструє застосування зазначених способів адресації: `MOV @R1,23h`. Тут виконується пересилка вмісту комірки з адресою 23*h* (пряма адресація) в комірку ОЗП, адреса якої записана в *R*1 (непряма регістрова адресація).

Область регістрів спеціальних функцій SFR містить акумулятор *ACC*, регістр-розширювач акумулятора *B*, покажчик стека *SP*, слово стану програми *PSW*, регістр-покажчик даних *DPTR*, регістри таймерів-лічильників, буфер приємопередавача і регістри управління. Регістри *SFR* допускають тільки пряму адресацію. Адреси регістрів *SFR* приведені на рис. 4.3, причому регістри, адреси яких помічені позначкою (*), допускають пряму адресацію окремих біт. Карта розподілення адрес регістрів *SFR* наведена на рис. 4.9.

Зовнішня пам'ять даних реалізується за допомогою додаткових мікросхем пам'яті і може мати ємність до 64 Кб з адресами 0 – *FFFFh*. Адресний простір резидентної і зовнішньої пам'яті даних не перетинаються, оскільки доступ до зовнішньої пам'яті здійснюється за допомогою спеціальних команд *MOVX*.

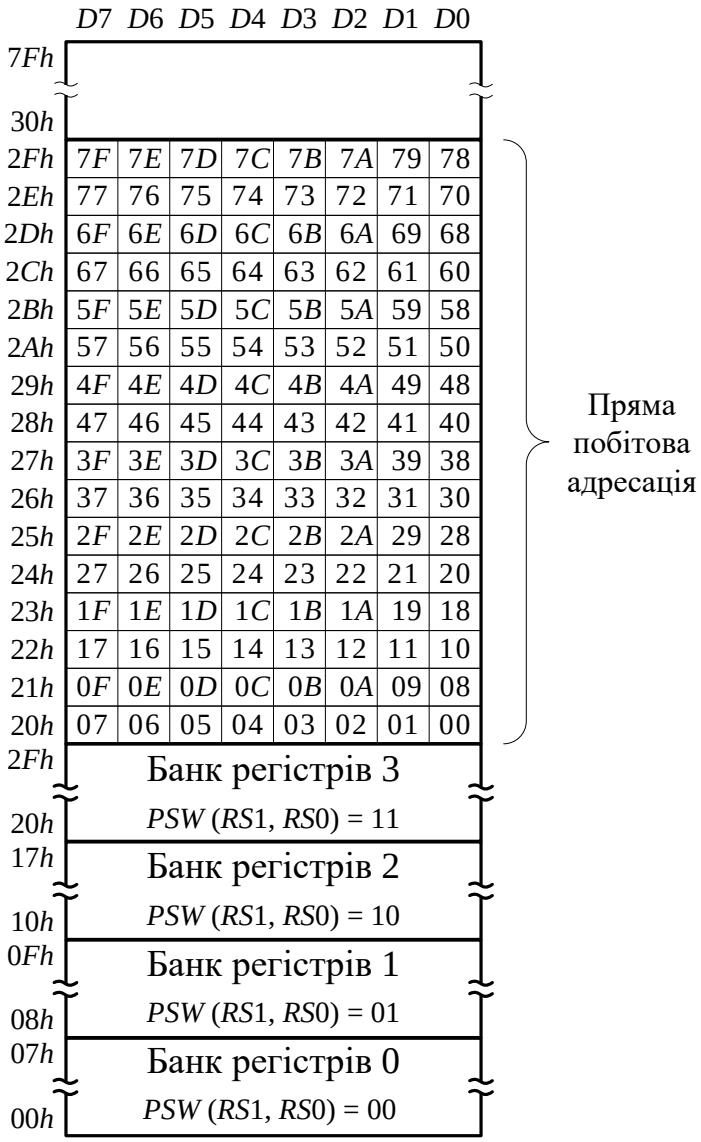


Рис. 4.8. Розподілення адрес ОЗП резидентної пам’яті даних

Звернення до зовнішньої пам'яті даних здійснюється тільки з використанням непрямої адресації, із застосуванням регістрів *R*1, *R*0 (команди `MOVX A,@Rr` і `MOVX @Rr,A`) або регістру-покажчика

даних (команди `MOVX A,@DPTR` і `MOVX @DPTR,A`). У першому випадку формуватиметься восьмирозрядна адреса, в другому – шістнадцятирозрядна. Видача адреси і передавання байта даних здійснюється аналогічно зовнішній пам'яті програм через порти *P*0 і *P*2. Під час звернення до зовнішньої пам’яті даних молодший байт адреси видається на порт *P*0 і старший байт – на порт *P*2 мікроконтролера. Обмін байтом даних (запис або зчитування) відбувається через порт *P*0 мікроконтролера, що працює як шина адреси/даних у режимі мультиплексування.

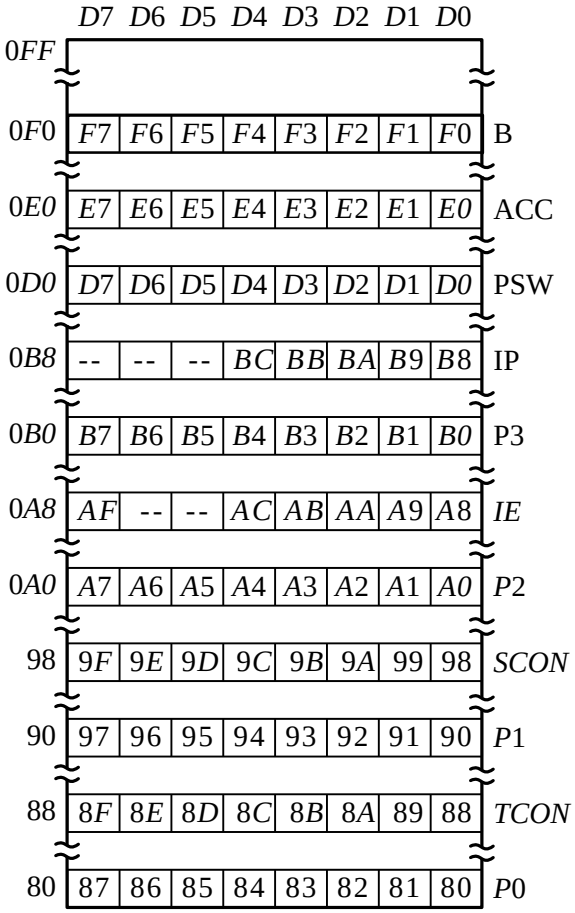


Рис. 4.9. Розподілення адрес блоку регістрів спеціальних функцій

Зчитування даних із зовнішньої пам'яті даних у мікроконтролер проводиться за допомогою вихідного сигналу *RD*, а запис даних у зовнішню пам'ять даних за допомогою вихідного сигналу *WR*.

На рис. 4.10 показана схема підключення зовнішньої пам'яті даних. Приведена схема дозволяє працювати з пам'яттю даних ємністю 2 Кб, використовуючи команди `MOVX A,@DPTR` і `MOVX @DPTR,A`. Порт *P*0 при цьому працює як мультиплексна шина адреса/дані, а три лінії порту *P*2 адресують сторінки зовнішнього ОЗП. Решта 5 ліній порту *P*2 використовуються в якості ліній вводу/виводу.

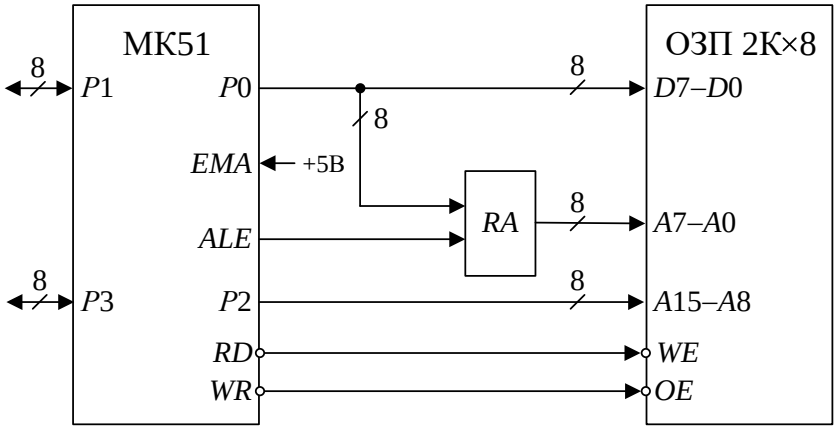


Рис 4.10. Підключення зовнішньої пам'яті даних ємністю 2 Кб

4.2.4. Таймери/лічильники

Два шістнадцятирозрядні таймери-лічильники – Т/Л1 і Т/Л0, призначено для отримання програмно керованих часових затримок і підрахунку зовнішніх подій. Кожний з них складається з двох восьмирозрядних регістрів: старшого – *THi* і молодшого – *TLi*, де $i = \overline{0,1}$ (рис. 4.3).

Під час роботи в якості таймера, в кожному машинному циклі виконується інкремент вмісту таймера/лічильника з частотою $f_Ч/12$, де $f_Ч$ – частота тактового генератора, оскільки машинний цикл складається з дванадцяти періодів частоти синхронізації.

Під час роботи в якості лічильнику вміст таймера/лічильника інкрементується за переходу з 1 в 0 зовнішнього сигналу, що

подається на вхід лічильника зовнішніх подій *T*1 (*T*0) мікросхеми МК51. Максимальна частота зовнішнього вхідного сигналу складає $f_Ч/24$.

Для управління режимами роботи таймера/лічильника і для організації взаємодії таймерів з системою переривань використовуються два регістри спеціальних функцій *TMOD і TCON*. Формати цих регістрів зображені на моделі програміста на рис. 4.3.

Регістр режимів таймера TMOD

GATE	$(TMOD.7)_{Т/Л1}$ $(TMOD.3)_{Т/Л0}$	– управління блокуванням, за встановлення розряду *GATE* = 1 дозволяється управляти Т/Л*i*, якщо зовнішній управляючий сигнал *INTi* = 1 і біт управління *TRi* встановлений, інакше управління Т/Л*i* дозволяється, тільки-но встановлюється біт управління *TRi*.
C/T	$(TMOD.6)_{Т/Л1}$ $(TMOD.2)_{Т/Л0}$	– біт вибору режиму, якщо *C/T* = 0, визначає роботу в якості таймера від внутрішнього джерела сигналів синхронізації, якщо *C/T* = 1, працює як лічильник від зовнішніх сигналів на вході *Ti*.
*M*1	$(TMOD.6)_{Т/Л1}$ $(TMOD.2)_{Т/Л0}$	– визначають режими 0 – 3 роботи таймера/лічильника
*M*0	$(TMOD.6)_{Т/Л1}$ $(TMOD.2)_{Т/Л0}$	

Розряди *M*0 і *M*1 регістра режимів таймера *TMOD* визначають чотири режими роботи Т/Л*i*:

***M*0**	***M*1**	**Режими роботи Т/Л*i***
0	0	Режим 0
0	1	Режим 1
1	0	Режим 2
1	1	Режим 3

Регістр управління/статусу таймера TCON

Призначення розрядів регістру управління/статусу таймера *TCON:*

*TF*1 (*TCON*.7) – *TF*0 (*TCON*.5) – ознака переповнювання таймерів Т/Л*i*, встановлюються програмно, або апаратно під час переповнювання Т/Л*i*; якщо переривання від відповідного таймера/лічильника дозволене, установка ознаки викличе переривання; ознаки скидаються програмно, або апаратно за обслуговування відповідного переривання;

*TR*1 (*TCON*.6) – *TR*0 (*TCON*.4) – біти управління таймерів Т/Л*i*, встановлюються і скидаються програмно;

Інші чотири біта регістра *TCON* призначені для управління перериваннями від зовнішніх сигналів *INT*1 і *INT*0.

*IE*1 (*TCON*.3) – *IE*0 (*TCON*.1) – ознаки запиту (фронту) зовнішніх переривань 1(0), Встановлюються апаратно за зрізом зовнішніх сигналів ЗПР1 (ЗПР0) або програмно. Скидаються апаратно при обслуговуванні переривання, викликаного фронтом сигналу переривання;

*IT*1 (*TCON*.2) – *IT*0 (*TCON*.0) – біти управління типом переривання 1(0), на входах *INT*1 і *INT*0; Встановлюються і скидаються програмно для специфікації запиту ЗПР1 (ЗПР0); якщо *ITi* = 0, то дозволено переривання за низьким рівнем сигналу, за встановлення *ITi* = 1 переривання за зрізом сигналу або за його низьким рівнем.

Режим 0 роботи таймера/лічильника

Логіка роботи Т/Л1 в режимі 0 показаний на рис. 4.11, *а*. Таймер/лічильник є тринадцятирозрядним лічильником, де послідовно з'єднані п'ятирозрядний регістр *TL*1 і восьмирозрядний регістр *TH*1. Залежно від значення розряду *C/T*1 регістра *TMOD* на вхід лічильника поступають зовнішні сигнали з входу *T*1 (лічильник) або сигнал *fч*/12 (таймер). Рахування розпочинається за встановлення біта *TR* регістра *TCON*. Управління рахуванням ззовні здійснюють за допомогою біту *GATE* регістра *TMOD*. При цьому рахувування дозволене за встановлення значення вхідного сигналу *INT*1 = 1 і заборонене за сигналом – *INT*0 = 0. Під час переповнювання Т/Л встановлюється ознака *TF*1.

Режим 1 роботи таймера/лічильника

Аналогічний режиму 0 з тією лише різницею, що Т/Л є шістнадцятирозрядним лічильником, тобто регістр *TL* – восьмирозрядний.

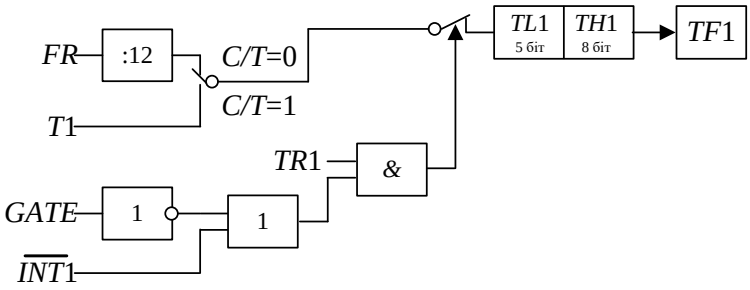
Режим 2 роботи таймера/лічильника

Приклад роботи Т/Л0 в режимі 2 показаний на рис. 4.11, *б*. Таймер/лічильник в такому режимі є восьмирозрядним лічильником на основі регістру *TL*0. Під час кожного переповнювання регістру *TL*0 відбувається завантаження вмісту регістру *TH*0 в регістр *TL*0. Вміст регістру *TH*0 завантажується програмно і в процесі рахування не змінюється.

Розглянуті режими 0, 1, 2 ідентичні для обох таймерів/лічильників.

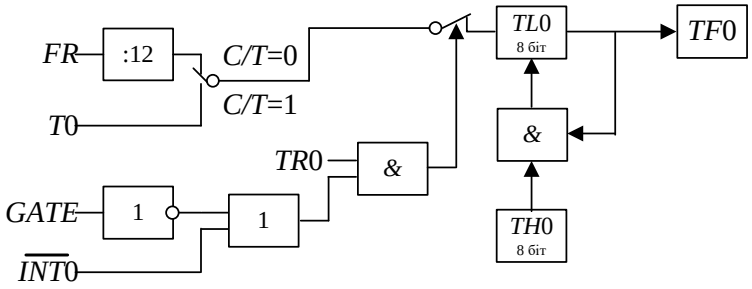
Режим 3 роботи таймера/лічильника

У режимі 3 робота Т/Л0 і Т/Л1 відрізняється. Таймер/лічильник Т/Л0 являє собою два незалежних пристрої – на основі регістру *TL*0 може працювати і як таймер, і як лічильник (рис. 4.11, *в*); Т/Л1, на основі регістру *TH*0, працює тільки в режимі таймера. Для включення останнього використовується біт *TR*1, під час переповнювання встановлюється ознака *TF*1. Таймер/лічильник Т/Л1 включений постійно, його біт *TR*1 встановлений, і працює в режимах 0, 1 або 2, не виставляючи ознаки переповнювання. Т/Л1 може бути використаний в будь-якому режимі, що не вимагає переривань. Наприклад, для роботи з послідовним інтерфейсом, який супроводжується сигналами переповнювання Т/Л1.

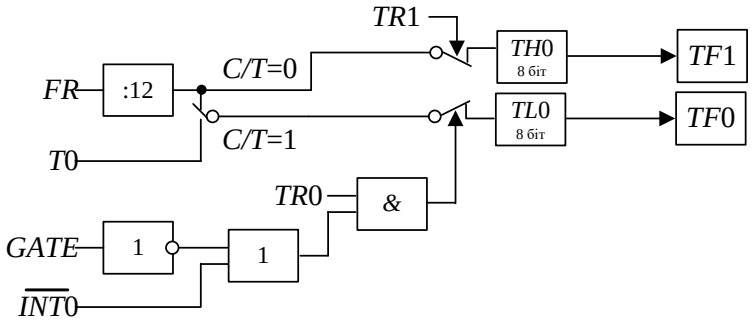


а

Рис. 4.11. Режими роботи таймера/лічильника: а – режим 0; б – режим 2; в – режим 3



б



в

Продовження рис. 4.11.

4.2.5. Порти вводу/виводу

Мікроконтролер містить чотири порти вводу/виводу: *P*0, *P*1, *P*2 *і P*3, які можуть бути використані для організації вводу/виводу інформації за тридцяти двома лініями передачі. Порт *P*0 є двоспрямованим, а порти *P*1, *P*2, *P*3 – квазідвоспрямованими. Їх особливість полягає в тому, що під час вводу над вхідними даними і поточним станом порту – даними, які виводилися з порту останніми, виконується порозрядна логічна операція І. Вихідні дані в порту запам'ятовуються.

Окрім виконань функцій вводу/виводу порти *P*0 – *P*3 можуть виконувати додаткові функції:

– через порт *P*0 видається молодший байт адреси і передається байт даних при роботі із зовнішньою пам'яттю програм і даних, а

також передаються дані під час програмування внутрішнього ППЗУ і зчитується його вміст;

– через порт *P*1 видається молодший байт адреси під час програмування внутрішнього ППЗУ і зчитування його вмісту.

– через порт *P*2 видається старший байт адреси даних при роботі із зовнішньою пам'яттю програм і даних, а також видається старший байт адреси при програмуванні внутрішнього ППЗУ і зчитуванні його вмісту.

– кожна лінія порту *P*3 має альтернативну функцію, на рис.4.3 наведене призначення відповідних виводів. Зазначені функції реалізуються, якщо у відповідному розряді регістра-заскочки записана одиниця.

Команди зчитування портів діляться на дві групи – команди, що зчитують інформацію з регістрів-заскочок і команди, що зчитують інформацію із зовнішніх контактів виходів портів. Перші група команд реалізує режим "зчитування-модифікація-запис", під час якого вміст регістра-заскочки зчитується, за необхідності модифікується і знов записується в регістр-заскочку. До таких команд належать команди `ANL`, `ORL`, `XRL`, `JBC`, `CPL`, `INC`, `DEC`, `DJNZ`, а також команди типа `MOV P1.1,C`, `CLR P2.5`, `SETB P2.7` (детальний опис команд наведений у розділі 4.3).

4.2.6. Блок послідовного інтерфейсу і переривань

Блок послідовного інтерфейсу і переривань призначений для вводу і виводу послідовної інформації і організації системи переривання програм. У його склад входять універсальний асинхронний приймач/передавач УАПП, буфер приймача/передавача *SBUF*, регістр управління/статусу УАПП *SCON*, регістри масок переривань *IE* і пріоритетів переривань *IP*.

Універсальний асинхронний приймач/передавач, який називають також послідовним портом, призначений для обміну інформацією, поданою послідовним кодом, і може працювати в наступних чотирьох режимах:

Режим 0 роботи УАПП

Інформація передається і приймається через зовнішній вхід приймача *RXD*. Через зовнішній вихід передавача видаються імпульси синхронізації, що стробують кожен біт інформації. Формат посилки – вісім біт, частота передачі і прийому – *F*/12.

У режимах 1 – 3 інформація приймається через вхід *RXD*, а передається через вихід *TXD*.

Режим 1 роботи УАПП

Формат посилки – десять біт: старт-біт (0), вісім біт даних і стоп-біт (1). Частота прийому і передачі задається Т/Л1. В цьому випадку необхідно заборонити переривання від Т/Л1 і запустити його на рахування в режимах 0, 1, 2. Найчастіше для цього використовується режим 2 таймера з автозавантаженням. В цьому випадку частота обміну даними визначається як

$$F = (2SMOD.fr)/(384.[256 - TH1]) \qquad (4.1)$$

де *fr* – частота тактового генератора складає.

Режим 2 роботи УАПП

Формат посилки – одинадцять біт: старт-біт (0), вісім біт даних, програмований дев'ятий біт і стоп-біт (1). Дев'ятий біт приймає значення бита *TB*8 регістра *SCON*. Частота прийому і передачі задається програмно значенням бита *SMOD* регістра *PCON* дорівнює *fr*/32, за *SMOD* = 1 або *fr*/64 за *SMOD* = 0.

Режим 3 роботи УАПП

Аналогічний режиму 2 за винятком того, що частота прийому і передачі задається Т/Л1, як в режимі 1.

Управління роботою УАПП здійснюється за допомогою регістра *SCON,* формат якого показаний на моделі програміста на рис. 4.3.

Регістр управління/статусу SCON

Призначення розрядів регістра управління/статусу *SCON*:

*SM*0 (*SCON*.7) – біти управління режими роботи УАПП, *SM*1 (*SCON*.6) встановлюються та скидаються програмно;

*SM*1 (*SCON*.5) – біт управління режимом УАПП, встановлюється програмно для заборони прийому повідомлення, в якому дев'ятий біт має значення нуля;

REN (*SCON*.4) – біт дозволу прийому, встановлюється і скидається програмно для дозволу і заборони прийому даних;

*TB*8 (*SCON*.3) – передача біта 8, дев'ятий біт даних, що передаються, в режимах 2, 3; встановлюється і

скидається програмно;

*RB*8 (*SCON*.2) – прийом біта 8, дев'ятий біт даних, що приймаються, в режимах 2, 3; у режимі 1 за встановлення *SM*2 = 0 є прийнятим стоп-бітом;

TI (*SCON*.1) – ознака переривання передавача; встановлюється апаратно за закінчення передачі байта; скидається програмно;

RI (*SCON*.0) – ознака переривання приймача; встановлюється апаратно за закінчення прийому байта; скидається програмно.

Розряди *SM*0 і *SM*1 регістра управління/статусу *SCON* визначають чотири режими роботи УАПП:

***SM*0**	***SM*1**	**Режими роботи УАПП**
0	0	Режим 0
0	1	Режим 1
1	0	Режим 2
1	1	Режим 3

За значення *SM*2 = 1 у режимах 2, 3 ознака *RI* регістра *SCON* не встановлюється, якщо прийнятий дев'ятий біт даних дорівнює нулю. Дев'ятий біт дозволяє вирішити задачу обміну інформацією між декількома мікроконтролерами, об'єднаними в локальну мережу за допомогою моноканалу. Наприклад, ведені (підлеглі) мікроконтролери встановлюють біти *SM*2 своїх регістрів *SCON* і продовжують виконувати програми. Головний мікроконтролер, якщо йому необхідно обмінятися інформацією з одним із ведених мікроконтролерів, посилає в моноканал байт-ідентифікатор абонента з одиничним дев'ятим бітом (встановлений біт *TB*8). Отримання такого байта викличе переривання ведених мікроконтролерів, які виконують підпрограму порівняння отриманого байта з кодом власної адреси. Мікроконтролер, який розпізнав адресу, скидає біт *SM*2 регістру *SCON* і готується до прийому даних. Решта мікроконтролерів не міняє значення цього біта і продовжує виконувати основну програму. Головний скидає біт *TB*8 і передає дані в моноканал.

У режимі 1 біт *SM*2 використовується для контролю істинності стоп-біта. Ознака *RI* не буде встановлена, якщо за встановлення *SM*2 = 1 не прийнятий стоп-біт, дорівнює одиниці.

Регістр управління потужністю PCON

Формат регістра *PCON* визначається серією мікроконтролера. В серії КР1816 регістр *PCON* має тільки один біт *SMOD*, що управляє швидкістю передачі послідовного порту. Якщо зазначений біт встановлений, швидкість передачі подвоюється.

В серії КР1830 регістр *PCON* додатково містить чотири біта (див. модель програміста на рис. 4.3). Біти *GF*1, *GF*0 (*PCON*.3, *PCON*.2) користувач може використовувати на свій розсуд. Встановлення біта *IDL* (*PCON*.0) приводить до переходу мікроконтролера в режим холостого ходу, коли блокуються функціональні вузли арифметико-логічного блоку і вузли, пов'язані з вибіркою команд, що зменшує енергоспоживання. При цьому зберігаються стани регістрів спеціальних функцій і РПД. Режим холостого ходу закінчується при активізації будь-якого дозволеного переривання або при апаратному скиданні по входу *RST*.

При встановленні біта *PD* (*PCON*.1) мікроконтролер переходить в режим мікроспоживання. Тактовий генератор вимикається, що приводить до припинення роботи всіх вузлів. При цьому зберігається вміст РПД. Виходом з такого режиму є апаратне скидання.

4.2.7. Система переривань

Система переривань дозволяє автоматично реагувати на зовнішні і внутрішні події. Джерелами переривання є зовнішні сигнали *INT*0 і *INT*1, таймери/лічильники та послідовний порт.

Переривання від зовнішніх сигналів *INT*1 і *INT*0 можуть бути ініційовані за зрізом сигналу або за його низьким рівнем, що визначається бітами *IT* регістра *TCON*. Переривання викликає встановлення одного з бітів *IE* регістра *TCON*. Переривання від таймерів/лічильників викликаються встановленням бітів *TF* регістра *TCON*, під час переповнювання лічильників. Переривання від послідовного порту УАПП викликаються встановленням бітів переривання приймача *RI* або передавача *TI* регістра *SCON*.

Переривання від кожного джерела може бути дозволено (заборонено) встановленням (скиданням) відповідного біта в регістрі масок переривань *IE* (див. модель програміста на рис. 4.3).

Регістр масок переривань IE

Призначення розрядів регістра масок переривань *IE*:

EA	(*IE*.7)	– за *EA* = 0 забороняє всі переривання; за *EA* = 1 переривання можуть бути дозволені бітами *EX*0, *ET*0, *EX*1, *ET*1, *ES* регістра *IE*, переривання дозволене якщо відповідний біт дорівнює одиниці;
	(*IE*.6)	– не використовується
	(*IE*.5)	– не використовується
ES	(*IE*.4)	управління перериванням від послідовного порту;
*ET*1	(*IE*.3)	– управління перериванням від Т/Л1;
*EX*1	(*IE*.2)	– управління перериванням від зовнішнього сигналу *INT*1;
*ET*0	(*IE*.1)	– управління перериванням від Т/Л0;
*EX*0	(*IE*.0)	– управління перериванням від зовнішнього сигналу *INT*0;

Система пріоритетів переривань є двохступінчастою. Кожному джерелу переривання може бути привласнений один з рівнів пріоритету – високий або низький, що визначається відповідним бітом регістра пріоритетів переривань *IP* (див. модель програміста на рис. 4.3). Призначення розрядів регістра *IP*:

	(*IP*.7) – (*IP*.5)	– не використовуються
PS	(*IP*.4)	– встановлення рівня пріоритету переривання від послідовного порту.
*PT*1	(*IP*.3)	– встановлення рівня пріоритету переривання від *Т/Л*1;
*PX*1	(*IP*.2)	– встановлення рівня пріоритету переривання від зовнішнього сигналу *INT*1;
*PT*0	(*IP*.1)	– встановлення рівня пріоритету переривання від *Т/Л*0;
*PX*0	(*IP*.0)	– встановлення рівня пріоритету переривання від зовнішнього сигналу *INT*0;

Наявність у відповідному розряді регістра *IP* одиниці встановлює для джерела високий рівень пріоритету, нуля – низький.

Програма обробки переривання з низьким рівнем пріоритету може бути перервана запитом переривання з високим рівнем пріоритету, але не може бути перервана іншим запитом з низьким рівнем пріоритету. Програма обробки переривання з високим рівнем пріоритету не може бути перервана ніяким іншим запитом переривання. Під час одночасного надходження двох запитів спочатку буде обслуговуватись переривання з високим пріоритетом. Якщо надійшли декілька запитів з однаковим рівнем пріоритету, обробка їх проводиться в порядку, визначеному послідовністю опитувань ознак переривань. Схема переривань МК51 зображена на рис. 4.12.

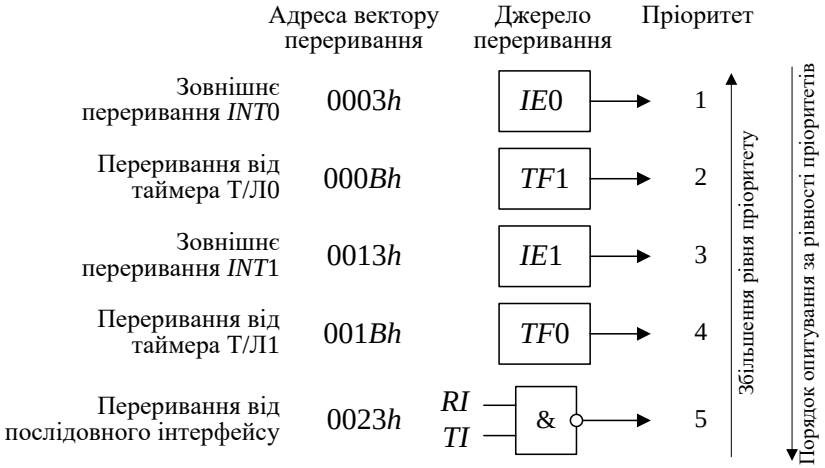


Рис. 4.12. Схема переривань МК51

4.3. Система команд мікроконтролера КР1816ВЕ51

4.3.1. Формат команд

Система команд мікроконтролера КР1816ВЕ51 значно ширша і потужніша ніж система команд КР1816ВЕ48, за рахунок команд множення, ділення, віднімання, операцій над бітами, операцій зі стеком, розширеного набору команд передачі управління.

Система команд мікроконтролера містить сто одинадцять команд. Відносно функціональних ознак команди класифікуються за наступними групами:

– команди передачі даних;
– команди виконання арифметичних операцій;
– команди виконання логічних операцій;
– команди виконання операцій з бітами;
– команди передачі управління.

Команди МК51 мають довжину один, два або три байти і виконуються відповідно за один, два або чотири машинні цикли. За тактової частоти генератора *fr* = 12Мгц тривалість циклу складає 1 мкс.

Можливі тринадцять форматів команд мікроконтролера приведено на рис. 4.13. Перший байт містить код операції (КОП), другий і третій – адреси операндів або безпосередньо операнди.

Операнди можуть бути чотирьох типів:

– *однобітні операнди* (біти), в якості яких можливо застосовувати окремі біти регістрів спеціальних функції *SFR* та портів; для адресації бітів застосовується пряма восьмибітна адреса (*bit*); непряма адресація бітів неможлива; карти адрес окремих бітів наведені на рис. 4.8 та рис. 4.9;

– *чотирибітні операнди* (тетради) застосовуються під час виконання операцій обміну тетрадами *SWAP* та *XCHD*;

– *восьмибітні операнди* (байти), являють собою комірки пам'яті програм або даних, константи – безпосередні операнди, регістри спеціальних функцій, порти вводу/виводу; порти та регістри спеціальних функцій *SFR* адресуються тільки прямим способом; байти пам'яті можуть адресуватися і непрямим способом, за допомогою адресних регістрів *R*0, *R*1, *DPTR, PC*.

– *двобайтні операнди* це константи та прямі адреси, для подання яких застосовуються другий і третій байти команди.

Система команд приведена в табл. 4.2. У табл. 4.2 і на рис. 4.13 використовуються наступні позначення:

#d – безпосередній операнд (восьмирозрядне двійкове число);
#d16 – безпосередній операнд (шістнадцятирозрядне двійкове число);
ad – восьмирозрядна адреса комірки РПД або регістрів спеціальних функцій, що прямо адресується;
ads, add адреса приймача, джерела відповідно;
*ad*11 одинадцятирозрядна адреса;

*ad*16 – шістнадцятирозрядна адреса;
bit – адреса біта
rel – восьмирозрядний зсув із знаком;
Rr – регістр з номером *r* (*r* = 0 – 7);
@Rr – операнд, що адресується способом непрямої адресації через регістр *Rr* при *r* = 0, 1 (комірки РПД або ЗПД);
(*X*) – вміст елементу *X*;
((*X*)) – вміст за адресою, що зберігається в елементі *X*;
(*X*)[15..8] – група розрядів елементу *X*;
\+ – операція додавання;
– – операція віднімання;
* – операція множення;
/ – операція ділення;
OR – операція логічне додавання АБО;
AND – операція логічне множення І;
XOR – операція ВИКЛЮЧНЕ АБО;
a[*j*] – розряди адреси;
A[*j*] – розряди акумулятора.

Номер формату команди	Байт1		Байт2		Байт3	
	*D*7	*D*0	*D*7	*D*0	*D*7	*D*0
1	КОП					
2	КОП		*#d*			
3	КОП		*ad*			
4	КОП		*bit*			
5	КОП		*rel*			
6	*a*[10] *a*[9] *a*[8] КОП		*a*[7] ... *a*[0]			
7	КОП		*ad*		*#d*	
8	КОП		*ad*		*rel*	
9	КОП		*ads*		*add*	
10	КОП		*#d*		*rel*	
11	КОП		*bit*		*rel*	
12	КОП		*ad*16[15-8]		*ad*16[7-0]	
13	КОП		*#d*16[15-8]		*#d*16[7-0]	

Рис. 4.13. Формати команд мікроконтролера

Таблиця 4.2. Система команд мікроконтролера КР1816ВЕ51

Мнемоніка	**КОП**	**Кількість циклів**	**Номер формату**	**Коментарі**
1	2	3	4	5
Команди передачі даних				
`MOV A,Rr`	`11101rrr`	1	1	Пересилка в акумулятор вмісту регістра; $(A) \leftarrow (Rr)$, $r = (7-0)$
`MOV A,ad`	`11100101`	1	3	Пересилка в акумулятор прямо адресованого байта; $(A) \leftarrow (ad)$
`MOV A,@Rr`	`1110011r`	1	1	Пересилка в акумулятор байта РПД; $(A) \leftarrow ((Rr))$, $r = 1,0$
`MOV A,#d`	`01110100`	1	2	Завантаження в акумулятор константи; $(A) \leftarrow \#d$
`MOV Rr,A`	`11111rrr`	1	1	Пересилка в регістр вмісту акумулятора; $(Rr) \leftarrow (A)$, $r = (7-0)$
`MOV Rr,ad`	`10101rrr`	2	3	Пересилка в регістр прямо адресованого байта; $(Rr) \leftarrow (ad)$, $r = (7-0)$
`MOV Rr,#d`	`01111rrr`	1	2	Завантаження в регістр константи; $(Rr) \leftarrow \#d$, $r = (7-0)$
`MOV ad,A`	`11110101`	1	3	Пересилка вмісту акумулятора за прямою адресою; $(ad) \leftarrow (A)$

Продовження таблиці 4.2.

1	2	3	4	5
MOV ad,Rr	10001rrr	2	3	Пересилка вмісту регістра за прямою адресою; $(ad) \leftarrow (Rr)$
MOV add,ads	10000101	2	9	Пересилка прямо адресованого байта за прямою адресою; $(add) \leftarrow (ads)$
MOV ad,@Rr	1000011r	2	3	Пересилка байта РПД за прямою адресою; $(ad) \leftarrow ((Rr))$, $r = 1, 0$
MOV ad,#d	01110101	2	7	Пересилка константи за прямою адресою; $(ad) \leftarrow \#d$
MOV @Rr,A	1111011r	1	1	Пересилка в РПД вмісту акумулятора; $(Rr) \leftarrow (A)$, $r = 1, 0$
MOV @Rr,ad	1010011r	2	3	Пересилка в РПД прямо адресованого байта; $((Rr)) \leftarrow (ad)$, $r = 1, 0$
MOV @Rr,#d	0111011r	1	2	Пересилка в РПД константи; $((Rr)) \leftarrow \#d$, $r = 1, 0$
MOV DPTR,#d16	10010000	2	13	Завантаження покажчика даних; $(DPTR) \leftarrow \#d16$
MOVC A,@A+DPTR	10010011	2	1	Пересилка в акумулятор байта пам'яті програм; $(A) \leftarrow ((A) + (DPTR))$

Продовження таблиці 4.2.

1	2	3	4	5
MOVC A,@A+PC	10000011	2	1	Пересилка в акумулятор байта пам'яті програм; $(PC) \leftarrow (PC)+1;\ (A) \leftarrow ((A)+(PC))$
MOVX A,@Rr	1110001r	2	1	Пересилка в акумулятор байта ЗПД; $(A) \leftarrow ((Rr)),\ r=1,0$
MOVX A,@DPTR	11100000	2	1	Пересилка в акумулятор байта розширеної ЗПД; $(A) \leftarrow ((DPTR))$
MOVX @Rr,A	1111001r	2	1	Пересилка в ЗПД із акумулятора; $((Rr)) \leftarrow (A),\ r=1,0$
MOVX @DPTR,A	11110000	2	1	Пересилка в розширену ЗПД із акумулятора $((DPTR)) \leftarrow (A)$
PUSH ad	11000000	2	3	Завантаження даних в стек; $(SP) \leftarrow (SP)+1;\ ((SP)) \leftarrow (ad)$
POP ad	11010000	2	3	Виштовхування даних зі стека; $(ad) \leftarrow ((SP));\ (SP) \leftarrow (SP)-1$
XCH A,Rr	11001rrr	1	1	Обмін вмістом акумулятора й регістра; $(A) \leftrightarrow (Rr),\ r=7-0$
XCH A,ad	11000101	1	3	Обмін вмістом акумулятора й прямо адресованого байта; $(A) \leftrightarrow ad$

Продовження таблиці 4.2.

1	2	3	4	5
XCH A,@Rr	1100011r	1	1	Обмін вмістом акумулятора й байта РПД; $(A) \leftrightarrow ((Rr))$, $r = 1,0$
XCHD A,@Rr	1101011r	1	1	Обмін вмістом молодшої тетради акумулятора й молодшої тетради байта РПД; $(A[3-0]) \leftrightarrow ((Rr[3-0]))$, $r = 1,0$
Арифметичні команди				
ADD A,Rr	00101rrr	1	1	Додавання до вмісту акумулятора вмісту регістру; $(A) \leftarrow (A) + (Rr)$, $r = (7-0)$
ADD A,ad	00100101	1	3	Додавання до вмісту акумулятора прямо адресованого байта; $(A) \leftarrow (A) + (ad)$
ADD A,@Rr	0010011r	1	1	Додавання до вмісту акумулятора байта РПД; $(A) \leftarrow (A) + ((Rr))$, $r = 1,0$
ADD A,#d	00100100	1	2	Додавання до вмісту акумулятора константи; $(A) \leftarrow (A)+\#d$
ADDC A,Rr	00111rrr	1	1	Додавання до вмісту акумулятора вмісту регістра і переносу; $(A) \leftarrow (A) + (Rr) + (C)$, $r = (7-0)$
ADDC A,ad	00110101	1	3	Додавання до вмісту акумулятора прямо адресованого байта і переносу; $(A) \leftarrow (A) + (ad) + (C)$

Продовження таблиці 4.2.

1	2	3	4	5
ADDC A,@Rr	0011011r	1	1	Додавання до вмісту акумулятора байта РПД і переносу; $(A) \leftarrow (A)+((Rr))+(C)$, $r=1,0$
ADDC A,#d	00110100	1	2	Додавання до вмісту акумулятора константи і переносу; $(A) \leftarrow (A)+\#d+(C)$
DA A	11010100	1	1	Десяткова корекція вмісту акумулятора: якщо $((A[3-0])>9$ або $(AC)=1)$, то $(A[3-0]) \leftarrow (A[3-0])+6$; або якщо $((A[7-4])>9$ або $(C)=1)$, то $(A[7-4]) \leftarrow (A[7-4])+6$
SUBB A,Rr	10011rrr	1	1	Віднімання з вмісту акумулятора вмісту регістра та позики; $(A) \leftarrow (A)-(Rr)-(C)$, $r=(7-0)$
SUBB A,ad	10010101	1	3	Віднімання з вмісту акумулятора прямо адресованого байта і позики; $(A) \leftarrow (A)-(ad)-(C)$
SUBB A,@Rr	1001011r	1	1	Віднімання з вмісту акумулятора байта РПД і позики; $(A) \leftarrow (A)-((Rr))-(C)$, $r=1,0$

Продовження таблиці 4.2.

1	2	3	4	5
SUBB A,#d	10010100	1	2	Віднімання з вмісту акумулятора константи і ознаки переносу; $(A) \leftarrow (A) - \#d - (C)$
INC A	00000100	1	1	Інкремент вмісту акумулятора; $(A) \leftarrow (A) + 1$
INC Rr	00001rrr	1	1	Інкремент вмісту акумулятора; $(Rr) \leftarrow (Rr) + 1,\ r = (7-0)$
INC ad	00000101	1	3	Інкремент прямо адресованого байта; $(ad) \leftarrow (ad) + 1$
INC @Rr	0000011r	1	1	Інкремент байта РПД; $((Rr)) \leftarrow ((Rr)) + 1,\ r = 1,0$
INC DPTR	10100011	2	1	Інкремент покажчика даних; $(DPTR) \leftarrow (DPTR) + 1$
DEC A	00010100	1	1	Декремент вмісту акумулятора; $(A) \leftarrow (A) - 1$
DEC Rr	00011rrr	1	1	Декремент вмісту регістра; $(Rr) \leftarrow (Rr) - 1,\ r = (7-0)$
DEC ad	00010101	1	3	Декремент прямо адресованого байта; $(ad) \leftarrow (ad) - 1$
DEC @Rr	0001011r	1	1	Декремент байта РПД; $((Rr)) \leftarrow ((Rr)) - 1,\ r = 1,0$

Продовження таблиці 4.2.

1	2	3	4	5
MUL AB	10100100	4	1	Множення акумулятора на регістр *B;* $(B).(A) \leftarrow (B)*(A)$
DIV AB	10000100	4	1	Ділення акумулятора на регістр *B*; $(B).(A) \leftarrow (B)/(A)$
Логічні команди				
ANL A,Rr	01011rrr	1	1	Логічне І вмісту акумулятора і регістра; $(A) \leftarrow (A)\ AND\ (Rr)$, $r = (7-0)$
ANL A,ad	01010101	1	3	Логічне І вмісту акумулятора і прямо адресованого байта; $(A) \leftarrow (A) AND (ad)$
ANL A,@Rr	0101011r	1	1	Логічне І вмісту акумулятора і байта РПД; $(A) \leftarrow (A)\ AND\ ((Rr))$, $r = 1,0$
ANL A,#d	01010100	1	2	Логічне І вмісту акумулятора і константи; $(A) \leftarrow (A) AND \# d$
ANL ad,A	01010010	1	3	Логічне І прямо адресованого байта і вмісту акумулятора; $(ad) \leftarrow (ad) AND (A)$
ANL ad,#d	01010011	2	7	Логічне І прямо адресованого байта і константи; $(ad) \leftarrow (ad) AND \# d$
ORL A,Rr	01001rrr	1	1	Логічне АБО вмісту акумулятора і регістра; $(A) \leftarrow (A) OR (Rr)$, $r = (7-0)$

Продовження таблиці 4.2.

1	2	3	4	5
ORL A,ad	01000101	1	3	Логічне АБО вмісту акумулятора і прямо адресованого байта; $(A) \leftarrow (A) OR (ad)$
ORL A,@Rr	0100011r	1	1	Логічне АБО вмісту акумулятора і байта РПД; $(A) \leftarrow (A) OR ((Rr))$, $r = 1,0$
ORL A,#d	01000100	1	2	Логічне АБО вмісту акумулятора і константи; $(A) \leftarrow (A) OR \# d$
ORL ad,A	01000010	1	3	Логічне АБО прямо адресованого байта і вмісту акумулятора; $(ad) \leftarrow (ad) OR (A)$
ORL ad,#d	01000011	2	7	Логічне АБО прямо адресованого байта і константи; $(ad) \leftarrow (ad) OR \# d$
XRL A,Rr	01101rrr	1	1	ВИКЛЮЧНЕ АБО вмісту акумулятора і регістра; $(A) \leftarrow (A) XOR (Rr)$, $r = (7-0)$
XRL A,ad	01100101	1	3	ВИКЛЮЧНЕ АБО вмісту акумулятора і прямо адресованого байта; $(A) \leftarrow (A) XOR (ad)$
XRL A,@Rr	0110011r	1	1	ВИКЛЮЧНЕ АБО вмісту акумулятора і байта РПД; $(A) \leftarrow (A) XOR ((Rr))$, $r = 1,0$
XRL A,#d	01100100	1	2	ВИКЛЮЧНЕ АБО вмісту акумулятора і константи; $(A) \leftarrow (A) XOR \# d$

Продовження таблиці 4.2.

1	2	3	4	5
`XRL ad,A`	`01100010`	1	3	ВИКЛЮЧНЕ АБО прямо адресованого байта і вмісту акумулятора; $(ad) \leftarrow (ad) XOR (A)$
`XRL ad,#d`	`01100011`	2	7	ВИКЛЮЧНЕ АБО прямо адресованого байта і константи; $(ad) \leftarrow (ad) XOR \# d$
`CLR A`	`11100100`	1	1	Скидання акумулятора; $(A) \leftarrow 0$
`CPL A`	`11110100`	1	1	Інверсія акумулятора; $(A) \leftarrow \overline{(A)}$
`RL A`	`00100011`	1	1	Зсув акумулятора вліво циклічний; $(A[j+1]) \leftarrow (A[j])$, де $j = (0-6)$; $(A[0]) \leftarrow (A[7])$
`RLC A`	`00110011`	1	1	Зсув акумулятора вліво з переносом; $(A[j+1]) \leftarrow (A[j])$, де $j = (0-6)$; $(A[0]) \leftarrow (C)$, $(C) \leftarrow (A[7])$
`RR A`	`00000011`	1	1	Зсув акумулятора вправо циклічний; $(A[j]) \leftarrow (A[j+1])$, де $j = (0-6)$; $(A[7]) \leftarrow (A[0])$

Продовження таблиці 4.2.

1	2	3	4	5
RRC A	00010011	1	1	Зсув акумулятора вправо з переносом; $(A[j]) \leftarrow (A[j+1])$, де $j = (0-6)$; $(A[7]) \leftarrow (C)$, $(C) \leftarrow (A[0])$
SWAP A	11000100	1	1	Обмін вмістом тетрад акумулятора; $(A[0-3]) \leftrightarrow (A[4-7])$
Команди операцій с бітами				
CLR C	11000011	1	1	Скидання ознаки переносу; $(C) \leftarrow 0$
CLR bit	11000010	1	4	Скидання біта; $(b) \leftarrow 0$
CPL C	10110011	1	1	Інверсія ознаки переносу; $(C) \leftarrow \overline{(C)}$
CPL bit	10110010	1	4	Інверсія біта; $(b) \leftarrow \overline{(b)}$
SETB C	11010011	1	1	Встановлення ознаки переносу; $(C) \leftarrow 1$
SETB bit	11010010	1	4	Встановлення біта; $(b) \leftarrow 1$
ANL C,bit	10000010	2	4	Логічне І біта і ознаки переносу; $(C) \leftarrow (C) AND (bit)$
ANL C,/bit	10110000	2	4	Логічне І інверсії біта і ознаки переносу; $(C) \leftarrow (C)\ AND\ \overline{(bit)}$
ORL C,bit	01110010	2	4	Логічне АБО біта і ознаки переносу; $(C) \leftarrow (C) OR (bit)$

Продовження таблиці 4.2.

1	2	3	4	5
ORL C,/bit	10100000	2	4	Логічне АБО інверсії біта і ознаки переносу; $(C) \leftarrow (C)\, OR\, \overline{(bit)}$
MOV C,bit	10100010	1	4	Пересилка біта в перенос; $(C) \leftarrow (bit)$
MOV bit,C	10010010	2	4	Пересилка переносу в біт; $(bit) \leftarrow (C)$
Команди передачі управління				
LJMP ad16	00000010	2	12	Довгий перехід в межах повного об'єму пам'яті програм; $(PC) \leftarrow ad16$
AJMP ad11	a[10]a[9] a[8]00001	2	6	Відносний перехід в межах сторінки об'ємом 2 Кб; $(PC) \leftarrow (PC) + 2$; $(PC[10-0]) \leftarrow ad11$
SJMP rel	10000000	2	5	Короткий перехід в межах сторінки об'ємом 256 байт; $(PC) \leftarrow (PC) + 2$; $(PC) \leftarrow (PC) + rel$
JMP @A+DPTR	01110011	2	1	Непрямий перехід; $(PC) \leftarrow (A) + (DPTR)$
JZ rel	01100000	2	5	Перехід, якщо вміст акумулятора дорівнює нулю; $(PC) \leftarrow (PC) + 2$; якщо $(A) = 0$, то $(PC) \leftarrow (PC) + rel$
JNZ rel	01110000	2	5	Перехід, якщо вміст акумулятора не дорівнює нулю; $(PC) \leftarrow (PC) + 2$; якщо $(A) <> 1$, то $(PC) \leftarrow (PC) + rel$

Продовження таблиці 4.2.

1	2	3	4	5
JC rel	01000000	2	5	Перехід, якщо встановлено ознаку переносу; $(PC) \leftarrow (PC) + 2$; якщо $(C) = 1$, то $(PC) \leftarrow (PC) + rel$
JNC rel	01010000	2	5	Перехід, якщо не встановлено ознаку переносу; $(PC) \leftarrow (PC) + 2$; якщо $(C) = 0$, то $(PC) \leftarrow (PC) + rel$
JB bit,rel	00100000	2	11	Перехід, якщо біт встановлено; $(PC) \leftarrow (PC) + 3$; якщо $(b) = 1$, то $(PC) \leftarrow (PC) + rel$
JNB bit,rel	00110000	2	11	Перехід, якщо біт не встановлено; $(PC) \leftarrow (PC) + 3$; якщо $(b) = 0$, то $(PC) \leftarrow (PC) + rel$
JBC bit,rel	00010000	2	11	Перехід, якщо біт встановлено, і скидання біта; $(PC) \leftarrow (PC) + 3$; якщо $(b) = 1$, то $(PC) \leftarrow (PC) + rel$; $(b) \leftarrow 1$
DJNZ Rr,rel	11011rrr	2	5	Декремент вмісту регістра і перехід, якщо вміст регістра не дорівнює нулю; $(PC) \leftarrow (PC) + 2$; $(Rr) \leftarrow (Rr) - 1$; якщо $(Rr) <> 0$, то $(PC) \leftarrow (PC) + rel$, $r = (7 - 0)$

Продовження таблиці 4.2.

1	2	3	4	5
`DJNZ ad,rel`	`11010101`	2	8	Декремент прямо адресованого байта і перехід, якщо його вміст не дорівнює нулю; $(PC) \leftarrow (PC)+2$; $(ad) \leftarrow (ad)-1$; якщо $(ad) <> 0$, то $(PC) \leftarrow (PC)+rel$
`CJNE A,ad,rel`	`10110101`	2	8	Порівняння вмісту акумулятора і прямо адресованого байта, перехід, якщо вміст не рівний; $(PC) \leftarrow (PC)+3$; якщо $(ad) < (A)$, то $(PC) \leftarrow (PC)+rel$, $(C) \leftarrow 0$; якщо $(ad) > (A)$, то $(PC) \leftarrow (PC)+rel$, $(C) \leftarrow 1$;
`CJNE A,#d,rel`	`10110100`	2	10	Порівняння вмісту акумулятора і константи і перехід, якщо значення не рівні; $(PC) \leftarrow (PC)+3$; якщо $\#d < (A)$, то $(PC) \leftarrow (PC)+rel$, $(C) \leftarrow 0$; якщо $\#d > (A)$, то $(PC) \leftarrow (PC)+rel$, $(C) \leftarrow 1$;
`CJNE Rr,#d,rel`	`10111rrr`	2	10	Порівняння вмісту регістра і константи, перехід, якщо значення не рівні; $(PC) \leftarrow (PC)+3$; якщо $\#d < ((Rr))$, то $(PC) \leftarrow (PC)+rel$, $(C) \leftarrow 0$; якщо $\#d > ((Rr))$, то $(PC) \leftarrow (PC)+rel$, $(C) \leftarrow 1$; $r = 7-0$

Продовження таблиці 4.2.

1	2	3	4	5
CJNE @Rr,#d,rel	1011011r	2	10	Порівняння байта РПД і константи, перехід, якщо значення не рівні; $(PC) \leftarrow (PC)+3$; якщо $\#d < ((Rr))$, то $(PC) \leftarrow (PC)+rel$, $(C) \leftarrow 0$; якщо $\#d > ((Rr))$, то $(PC) \leftarrow (PC)+rel$, $(C) \leftarrow 1$; $r = 1, 0$
LCALL ad16	00010010	2	12	Довгий виклик підпрограми; $(PC) \leftarrow (PC)+3$; $(SP) \leftarrow (SP)+1$, $((SP)) \leftarrow (PC[7..0])$, $(SP) \leftarrow (SP)+1$, $((SP)) \leftarrow (PC[15..8])$, $(PC) \leftarrow ad16$
ACALL ad11	a[10]a[9] a[8]10001	2	6	Відносний виклик підпрограми в межах сторінки об'ємом 2 Кб; $(PC) \leftarrow (PC)+2$; $(SP) \leftarrow (SP)+1$, $((SP)) \leftarrow (PC[7..0])$, $(SP) \leftarrow (SP)+1$, $((SP)) \leftarrow (PC[15..8])$, $(PC)[10..0] \leftarrow ad11$
RET	00100010	2	1	Повернення з підпрограми; $(PC[15..8]) \leftarrow ((SP))$, $(SP) \leftarrow (SP)-1$, $(PC[7..0]) \leftarrow ((SP))$, $(SP) \leftarrow (SP)-1$

Продовження таблиці 4.2.

1	2	3	4	5
RETI	00110010	2	1	Повернення з переривання; $(PC[15..8]) \leftarrow ((SP))$, $(SP) \leftarrow (SP)-1$, $(PC[7..0]) \leftarrow ((SP))$, $(SP) \leftarrow (SP)-1$
NOP	00000000	1	1	Немає операції; $(PC) \leftarrow (PC)+1$
Примітки: ЗПД зовнішня пам'ять даних – ємність 256 байт, розширена ЗПД – ємність до 64 Кб.				

4.3.2. Способи адресації операндів

У мікроконтролері МК51 застосовуються наступні способи адресації джерел операндів:

– пряма адресація;
– безпосередня адресація;
– пряма регістрова адресація;
– пряма побітова адресація;
– непряма регістрова адресація;
– непряма адресація за сумою базового і індексного регістра.

Перші три із зазначених способів адресації використовуються для адресації приймачів операндів. Використовувані способи адресації, поєднані в різних комбінаціях, забезпечують двадцять один режим адресації.

Формати більшості команд містить поля: «приймач» і «джерело» операндів, які визначають тип даних, способи адресації і саме операнди, що приймають участь в обчисленнях.

В командах, що не виконують операції перезапису, операнд призначення є операндом-джерелом.

Велика кількість команд включає операнди, розташовані у внутрішньому ОЗП даних МК51. Вибір адресного простору пам'яті програм або зовнішньої пам'яті даних як другий операнд визначається командною мнемонікою (якщо тільки другий операнд не є безпосередньою величиною).

Область внутрішнього ОЗП даних, що адресується, визначається способом адресації і величиною адреси. Наприклад, звернення до регістрів спеціального призначення може бути виконане тільки за допомогою прямої адресації.

Пряма адресація

Пряма адресація використовується для звернення до комірок резидентної пам'яті даних ОЗП з адресами (0 – 127) і до регістрів спеціального призначення.

Команди з прямою адресацією мають довжину 2 байти. Перший байт призначений для коду команди, другий байт містить адресу елементу резидентної пам'яті даних, де зберігається операнд. Оскільки адресний простір внутрішньої пам'яті даних МК51 відповідає області *00Fh – 0FFh*, то у використанні двобайтного формату немає необхідності. Умовне позначення адреси в мнемоніці команди – *ad*.

***Приклад 4.1*:** Пряма адресація

```
ADD     A,ad        ; Запис команди додавання з прямою
                    адресацією ; в загальному вигляді.
MOV     ads,add     ; Запис команди пересилання даних з прямою
                    ; адресацією в загальному вигляді, коли вміст
                    ; елементу пам'яті з адресою ads пересилається
                    в ; елемент пам'яті з адресою add.
ADD     A,31h       ; Додавання вмісту акумулятора до вмісту
                    ; комірки резидентної пам'яті з адресою 31h.
                    ; Результат розміщується в акумуляторі.
MOV     21h,10h     ; Пересилання вмісту елементу пам'яті з
                    адресою ; 10h в елемент пам'яті з адресою 21h.
```

Безпосередня адресація

***Приклад 4.2*:** Безпосередня адресація

```
ADD     A,#d8           ; Запис в загальному вигляді команди
                        дода-
                        ; вання з безпосередньою адресацією.
ADD     A,#3            ; Додавання вмісту акумулятора i числа 3,
                        ; результат розміщується в акумуляторі.
ORL     A,#0Ah          ; Виконання операції порозрядного АБО
                        над ; вмістом акумулятора і числом 0Ah,
                        ; результат розміщується в акумуляторі.
MOV     DPTR,#3400h     ; Завантаження в регістр-покажчик DPTR
                        ; числа 3400h.
```

Всього одна команда з безпосередньою адресацією `MOV DPTR,#d16` має трибайтний формат, оскільки виконує завантаження даних в двобайтний регістр *DPTR*.

Пряма регістрова адресація

Пряма регістрова адресація використовується під час звернення до робочих регістрів одного з банків регістрів.

Пряма регістрова адресація використовується також для звернення до акумулятору *ACC*, регістрів *B, AB* (*AB* – використовується, як регістр подвійної довжини), регістру *DPTR* і до ознаки переносу *C*. Використання прямої регістрової адресації дозволяє отримувати двобайтний еквівалент команд довжиною три байти з прямою адресацією даних.

У командах з прямою регістровою адресацією один з операндів знаходиться в одному з регістрів загального призначення $Rr = R0..R7$ банку, номер якого визначається розрядами *RS*1, *RS*0 регістра ознак *PSW*. Якщо команда виконує дії над одним операндом, то формат команди однобайтний. Номер регістра *Rr* визначається трьома молодшими бітами коду операції.

***Приклад 4.3*:** Пряма регістрова адресація

```
MOV   A,Rr   ; Запис команди завантаження акумулятора з
             ; регістра Rr (пряма регістрова адресація) в
             ; загальному вигляді, де r = 7 – 0.
MOV   A,R5   ; Завантаження акумулятора операндом із
             ; регістра R5.
```

Двоадресні команди, які використовують пряму регістрову адресацію разом з безпосередньою або прямою адресацією, мають двобайтний формат команди.

***Приклад 4.4*:** Регістрова адресація разом із прямою та безпосередньою адресацією.

```
MOV   ad,Rr  ; Запис в загальному вигляді команди
             ; завантаження вмісту комірки пам'яті з адресою
             ; ad (пряма адресація) з регістра Rr (пряма
             ; регістрова адресація), де r = 7 – 0.
MOV   30h,R5 ; Завантаження комірки пам'яті з адресою 30h
             ; (пряма адресація) з регістра R5 (пряма
             ; регістрова адресація).
MOV   R7,#06 ; Завантаження в регістр R7 (пряма регістрова
             ; адресація) число 6 (безпосередня адресація).
```

Пряма побітова адресація

Пряма побітова адресація використовуються для звернення до прямо адресованих бітів комірок ОЗП з адресами 20*H* – 2*FH*. Вказані комірки послідовно пронумеровані від молодшого біта молодшого байта до старшого біта старшого байта (рис. 4.8).

Пряма побітова адресація використовуються до бітів регістрів спеціального призначення. Прямо адресовані біти в регістрах спеціального призначення пронумеровані таким чином: п'ять старших розрядів адреси співпадають з п'ятьма старшими розрядами

адреси самого регістра, а три молодших – визначають місцеположення окремого біта усередині регістра (рис. 4.9).

Непряма регістрова адресація

Непряма регістрова адресація використовується під час звернення до комірок резидентної пам'яті даних, при цьому як покажчики адреси використовуються регістри *R*0, *R*1 вибраного банку регістрів. У командах `PUSH` і `POP` в якості покажчика адреси використовується вміст покажчика стека *SP*.

Непряма регістрова адресація використовується під час звернення до зовнішньої пам'яті даних. В цьому випадку за застосування регістрів-покажчиків *R*0 і *R*1 (вибраного банка робочих регістрів) адресується комірка однієї із сторінок об'ємом 256 байт зовнішньої пам'яті даних. Номер сторінки заздалегідь задається вмістом одного з портів вводу/виводу, наприклад *P*1.

Під час звернення до будь-якої комірки адресного простору зовнішньої пам'яті даних об'ємом до 64 Кб застосовується шістнадцятирозрядний покажчик даних *DPTR*.

Команди з непрямою регістровою адресацією мають довжину 1 байт, де розташований код операції. Адреса операнда знаходиться в регістрі *R0* або *R*1 поточного банку регістрів, якщо обмін даними або операція виконуються у внутрішньому ОЗП даних.

***Приклад 4.5*:** Непряма регістрова адресація

```
ADD   A,@R0   ; Скласти вміст акумулятора ACC з вмістом
              ; комірки пам'яті, адреса якої знаходиться в
              ; регістрі R0.
DEC   @R      ; Зменшити на 1 вміст комірки пам'яті, адреса
              ; якої знаходиться в регістрі R1.
```

За застосування непрямої регістрової адресації під час звернення до зовнішньої пам'яті даних, адреса елементу зовнішньої пам'яті даних може бути двобайтною. Тому під час використання в якості покажчику молодшого байту адреси регістрів *R*0 або *R*1, старший байт адреси повинен бути заздалегідь завантажений в порт вводу/виводу *P*2. Окрім того, під час звернення до двобайтної адреси зовнішньої пам'яті даних в якості регістра-покажчика може бути використаний двобайтний регістр *DPTR*, при цьому звичайно попереднє формування старшого байта адреси в порту *P*2 не потрібне.

***Приклад 4.6*:** Непряма регістрова адресація за звернення до двобайтної адреси елементу зовнішньої пам'яті даних.

```
MOVX    A,@R0       ; Пересилання в акумулятор вмісту
                    ; комірки зовнішньої пам'яті даних,
                    ; старший байт адреси якої знаходиться в
                    ; порту P2, молодший байт – у регістрі R0
                    ; поточного банку регістрів.
MOVX    A,@DPTR     ; Пересилання в акумулятор вмісту
                    ; комірки зовнішньої пам'яті даних, старші
                    ; і молодші байти адреси якої знаходяться
                    ; в двобайтному регістрі DPTR.
```

Непряма регістрова адресація за сумою базового і індексного регістрів

Непряма регістрова адресація використовується для читання даних з пам'яті програм, зокрема для вибірки елементу з порядковим номером з таблиці.

Непряма регістрова адресація за сумою базового та індексного регістрів (вмісту акумулятора) спрощує перегляд таблиць, записаних в пам'яті програм. Будь-який байт з таблиці може бути вибраний за адресою, яка визначається сумою вмісту регістру *DPTR* або регістру *PC* і вмісту *A*.

***Приклад 4.7*:** Непряма регістрова адресація за сумою базового і індексного регістрів.

```
MOVC    A@A+DPTR    ; Пересилання в акумулятор вмісту
                    ; комірки пам'яті програм, адреса якої
                    ; обчислюється шляхом додавання
                    ; двобайтного значення регістра-покажчика
                    ; DPTR і однобайтного операнду без знаку,
                    ; розміщеного в акумуляторі.
                    ; Цю операцію інтерпретують, як читання
                    ; елементу з номером i, заданим в
                    ; акумуляторі, із таблиці, початкова адреса,
                    ; якої задана в DPTR.
MOVC    A@A+PC      ; Пересилання в акумулятор вмісту комірки
                    ; області пам'яті програм, адреса якої
                    ; дорівнює сумі поточного значення.
```

```
                    ; лічильника команд PC і значення
                    ; операнда в акумуляторі.
```

Відносна адресація

Відносна адресація використовується в командах умовних переходів, які слугують для організації розгалуження програм. Команди умовних переходів мають двобайтний або трибайтний формат. У командах типу «перейти за ознакою» перший байт містить код операції, а другий – зміщену адресу (або адреси) переходу відносно адреси поточної команди в форматі цілого числа із знаком. Діапазон можливих кодів зсуву: від –128 до +127. У командах типу «виконати операцію і перейти за результатом» перший байт містить код операції, другий байт – операнд для виконання заданої операції, третій байт – зміщену адресу переходу.

Якщо умова команди умовного переходу виконується, то адресу переходу процесор обчислює шляхом додавання поточної адреси і коду зміщення. Якщо умова не виконується, то МК51 переходить до виконання наступної команди.

Під час написання програм немає необхідності обчислювати абсолютні коди зміщення для команд умовного переходу. Досить вказати лише мітку, чисельне значення коду зміщення обчислить програма компілятор.

***Приклад 4.8*:** Відносна адресація

```
JZ      rel       ; Перейти на мітку rel, якщо значення
                  ; в акумуляторі дорівнює нулю.
DJNZ    ad,rel    ; Відняти одиницю з вмісту комірки пам'яті з
                  ; адресою ad і перейти на мітку, якщо
                  ; результат не дорівнює нулю.
```

4.3.3. Формування ознак результату

Слова стану програми *PSW* зберігає чотири ознаки результату виконання команд: *C* – переносу, *AC* –додаткового переносу *OV* – переповнювання, *P* – парності (паритету).

Ознака С встановлюється, якщо в старшому розряді результату виникає перенос або позика. Під час виконання операцій множення та ділення ознака С скидається.

Ознака *AC* встановлюється при виконанні додавання і віднімання, між тетрадами байта виникає перенос або позика.

Ознака *OV* встановлюється, коли результат додавання або віднімання виходить за межі семи розрядів і восьмий розряд не може трактуватися, як знаковий. Тобто ознака *OV* вказує на переповнювання розрядної сітки під час виконання операцій зі знаками. Ознака *OV* під час виконання операції множення встановлюється, якщо результат перевищує значення 255 (0*FFh).* Під час виконання операції ділення ознаки *OV* скидаються, коли дільник дорівнює нулю – встановлюється.

Ознака *P* залежить від поточного значення вмісту акумулятора. Якщо кількість одиничних біт непарне, ознака встановлюється, парне – скидається.

Ознаки *OV* і *P* відсутні у МК48.

Команди, виконання яких модифікує ознаки результату, наведені у табл. 4.3. Значення ознаки парності змінюється всіма командами, що змінюють вміст акумулятора. Окрім зазначених у табл. 4.3 команд ознаки змінюють команди, в яких призначення результату визначає регістр *PSW*, або його окремі біти, а також операції над бітами.

Таблиця 4.3. Формування ознак результату

Мнемоніка	**Ознаки**			**Мнемоніка**	**Ознаки**		
ADD	*C*	*OV*	*AC*	CLR C	0	–	–
ADDC	*C*	*OV*	*AC*	CPL C	*C*	–	–
SUBB	*C*	*OV*	*AC*	ANL C,bit	*C*	–	–
MUL	0	*OV*	–	ANL C,/bit	*C*	–	–
DIV	0	*OV*	–	ORL C,bit	*C*	–	–
DA	*C*	–	–	ORL C,/bit	*C*	–	–
RRC	*C*	–	–	MOV C,bit	*C*	–	–
RLC	*C*	–	–	CJNE	*C*	–	–
SETB C	1	–	–				

4.3.4. Виконання команд передачі даних

До групи команд передачі даних належать команди передачі та обміну байтами. Команди передачі даних не модифікують ознаки результату, окрім команди завантаження регістру PSW та акумулятора – ознака парності.

На відміну від МК48 у МК51 обмін даними може відбуватися і без участі акумулятора.

Взагалі можливі дев'ять операндів між якими відбувається обмін даними. Граф можливих операцій передачі даних наведений на рис. 4.14.

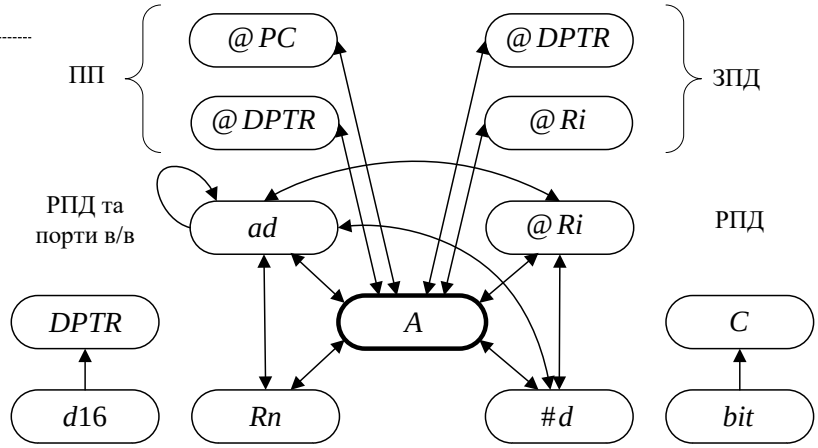


Рис. 4.14. Граф шляхів передачі даних

Команда завантаження покажчика даних завантажує в регістр *DPTR* двобайтну константу з другого і третього байтів команди. При цьому другий байт команди пересилається в регістр *DPH*, третій – в регістр *DPL*.

Зчитування даних із пам'яті програм виконується командами `MOVC`. Адреса байта, що зчитується, обчислюється як сума вмісту акумулятора (число без знаку) і одного з двох шістнадцятирозрядних регістрів (*DPTR* або *PC*).

Є два типи команд `MOVX` обміну із зовнішньою пам'яттю даних. Перший, забезпечує доступ до 256 комірок ЗПД, причому адреса розташовується в регістрах *R*1 або *R*0 поточного байта регістрів. Другий забезпечує доступ до розширеної ЗПД, яка може містити до 64 Кб пам'яті. Адреса в цьому випадку розташовується в регістрі-покажчику даних *DPTR*.

***Приклад 4.9*:** Передати вміст буфера УАПП в РДП. Застосувати непряму адресацію даних, адреса, розміщена в *R*0:

```
; програма
MOV    @ RO,SBUF     ; передавання, прийнятого за
                     ; послідовним каналом байта в РПД.
```

Приклад 4.10: Завантажити в покажчик даних початкову адресу *7F*00*h* масиву даних, розташованих в ЗДП:

```
; програма
MOV     DPTR,7FOOH      ; завантаження початкової
                        ; адреси масиву даних.
```

Приклад 4.11: Завантажити управляюче слово в регістр управління таймером:

```
; програма
MOV     TCON,00000101B    ; скидання області бітів.
```

Приклад 4.12: Скинути всі ознаки користувача в області бітів РПД, розташованої розпочинаючи з адреси 20*h* до адреси 2*Fh*

```
; програма
        MOV     R0,20H      ; завдання початкової адреси
                            ; області ознак
        MOV     R1,0FH      ; лічильник (довжина області
                            ; ознак)
LOOP:   MOV     @R0,#0      ; скидання одного байту (8 ознак)
        INC     R0          ; перехід до наступного байту
        DJNZ    R1,LOOP     ; кінець циклу
```

Приклад 4.13: Переслати в ЗПД вміст регістрів банка регістрів 1, початкова адреса області ЗПД – 5000*h*:

```
; програма
      MOV     PSW,#8H       ; вибір банку регістрів 1,
                            ; PSW[4,3] = 01
      MOV     R0,#8         ; лічильник
      MOV     DPTR,#5000H   ; завдання початкової адреси ЗПД
      MOV     R1,0          ; завдання початкової адреси РПД
L1:   MOV     A,@R1         ; пересилання (A) ← (R1)
      MOVX    @DPTR,A       ; пересилання з акумулятора в
                            ; ЗПД
      INC     R1            ; перехід до наступного регістру
      INC     DPTR          ; приріст покажчика адреси
```

```
DJNZ   R0,L1          ; R0= R0 – 1, якщо R0 > 0, то
                      ; повторити цикл
```

***Приклад 4.14*:** Реалізувати звернення до таблиці готових рішень, що зберігається в пам'яті програм для обчислення синусу кута *X*, якщо *X* змінюється в межах від 0 до 89 градусів із дискретністю один градус.

Для вирішення даної задачі доцільно скласти таблицю готових рішень і зберегти її у пам'яті програм. Для звернення до пам'яті програм застосовують спеціальні команди – MOVC. Найбільш швидке обчислення функції синусу кута X можна дістати шляхом вибірки готового значення синусу з таблиці. Така таблиця для діапазону 0 – 89 градусів займе 90 байтів при похибці 0,4%. Кожний байт таблиці буде містить дробову частину двійкового подання синусу. Вихідним параметром для програми служить значення кута *X*, яке знаходиться в акумуляторі. Наведемо програму:

```
; програма
; обчислення SIN(X) за таблицею значень
; у вихідному стані (A) ← (X, цілі числа від 0 до 89)
; в результаті отримуємо: (A) ← (дробова частина SIN(X))
SINX:   INC    A              ; інкремент акумулятора
        MOVC   A,@A+PC        ; завантаження значення
                              ; синусу з таблиці
        RET                   ; повернення
; формування таблиці значень синусу
SINUS:  DB     0              ; SIN (0) = 0
        DB     000001001B     ; SIN (1) = 0.017
        DB     00001001B      ; SIN (2) = 0.035
                              ; …
        DB     11111111B      ; SIN(89) = 0.999
```

Слід зазначити, що у програмі не використаний покажчик даних *DPRT*. Перед зверненням до таблиці інкрементують вміст акумулятору, що необхідно із-за наявності однобайтної команди повернення, розташованої між командою MOVC і початком таблиці значень синуса.

4.3.5. Виконання операцій зі стеком

Механізм доступу до стеку МК51 аналогічний МК48: перед завантаженням у стек вміст регістра-вказівника стека *SP* інкрементується, а після виштовхування зі стеку декрементується.

За сигналом системного скидання в *SP* заноситься початкове значення *07h*. Для перевизначення *SP* можна скористатися командою `MOV 81h,@d` (де 81*h* – адреса регістру *SP*).

Таким чином, стек може розташовуватись в будь-якому місці РПД. Стек використовується для організації звернень до підпрограм, при обробці переривань, для передачі параметрів підпрограм, для тимчасового зберігання вмісту регістрів спеціальних функцій.

Підпрограма обробки переривань повинна зберегти у стеку зміст тих регістрів, які вона сама буде використовувати, а перед поверненням у перервану програму повинна відновити їх значення.

***Приклад 4.15*:** Розробити підпрограму обробки зовнішнього переривання нульового рівня.

```
; підпрограма
        ORG    3          ; завдання адреса вектора переривання
        SJMP   SUB        ; перехід на підпрограму обробки
        ORG    3OH
SUB:    PUSH   PSW        ; збереження у стеку PSW
        PUSH   ACC        ; збереження акумулятора
        PUSH   B          ; збереження B
        PUSH   DPL        ; збереження DPL
        PUSH   DPH        ; збереження DPH
        ANL    D0h,F7h    ; PSW[4, 3] = 00
        MOV    PSW,8H     ; вибір банку регістрів 1
; обробка переривань
        POP    DPH        ; відновлення DPH
        POP    DPL        ; відновлення DPL
        POP    B          ; відновлення B
        POP    ACC        ; відновлення акумулятора
        POP    PSW        ; відновлення PSW і номеру
                          ; банку регістрів
        RETI              ; повернення з підпрограми
```

4.3.6. Виконання арифметичних операцій

До групи арифметичних команд належать операції множення, ділення, підсумовування, віднімання, тощо.

Під час виконання операції `MUL` множаться цілі без знаку, які у вихідному стані розміщуються в акумуляторі і в регістрі *B*. Старший байт результату формується в регістрі B, молодший – в акумуляторі. Ознака переповнювання встановлюється, якщо результат перевищує значення 255 (0*FFh).*

Команда ділення `DIV` дозволяє ділити цілі числа без знаку. У вихідному стані ділене розміщується в акумуляторі, дільник – в регістрі *B*. Після виконання операції ціла частина частки розміщується в акумуляторі, залишок – в регістрі *B*. Ознаки *C* і *OV* скидаються. Виняток становить випадок, коли дільник дорівнює нулю, при цьому ознака переповнювання встановлюється. У табл. 4.2 приведені команди, що впливають на встановлення ознак у регістрі *PSW*.

Під час виконання команд додавання `ADD` або додавання з переносом `ADDC` встановлюються ознаки *C* і *AC,* при виникненні переносів із 7 та 3 розрядів відповідно. Ознака *C* під час додавання чисел без знаку вказує на появу переповнювання розрядної сітки. Біт *OV* встановлюється, якщо є перенос із біта 6 і немає переносу з біта 7, або, якщо немає переносу з біта 6 і є перенос із біта 7. За додавання чисел із знаком біт *OV* указує на від'ємну суму при додаванні додатних операндів або на додатну суму двох від'ємних доданків.

У командах «Віднімання з позикою» `SUBB` ознаки *C* і *AC* встановлюються, якщо необхідна позика для бітів 7 і 3. Під час віднімання чисел із знаком біт *OV* свідчить про від'ємний результат, коли з додатного числа віднімають від'ємне, або про додатний результат, коли з від'ємного числа віднімають додатне.

Команда десяткової корекції `DA` корегує в акумуляторі результат додавання (командами `ADD` або `ADDC`) двох змінних, поданих в двійково-десятковому форматі. Ціль операції отримати дві двійково-десяткові цифри. Команда виконує двійково-десяткове перетворення шляхом додавання вмісту акумулятора з числами 06h, 60h, 66h залежно від початкового стану акумулятора і ознак *C, AC*. Команда `DA` застосовується тільки після операції додавання і не може використовуватися, наприклад, після операції віднімання, а

також виконувати просте перетворення шістнадцятирічного числа в двійково-десяткове.

Приклад 4.16. Скласти два двійкових багатобайтних числа. Обидва доданки розташовуються у РПД послідовно, розпочинаючи з молодшого байту. Початкові адреси доданків задані в *R*0 і *R*1. Формат доданків, що визначає кількість байтів задано в *R*2:

```
; програма
        CLR     C           ; скидання переносу
L1:     MOV     A,@R0       ; завантаження. в акумулятор
                            ; поточного байта першого доданку
        ADDC    A,@R1       ; складання байт з урахуванням
                            ; переносу
        MOV     @R0,A       ; розміщення байта результату
        INC     R0          ; просування покажчиків
        INC     R1
        DJNZ    R2,L1       ; цикл, якщо не всі байти додані
```

Під час додавання чисел без знаку на наявність переповнення вказує ознака *C*, а за додавання чисел із знаком – ознака *OV*. Час додавання складає (1+7*N*) мкс, де *N* – формат операндів у байтах.

Приклад 4.17. Виконати множення цілого двійкового багатобайтного числа і константи 173. Вихідне двійкове число розташовується в РПД, адреса молодшого байту зберігається у регістрі *R*0. Формат числа у байтах зберігається в регістрі *R*1:

Команда *MUL* у МК51 обчислює добуток двох цілих чисел без знаку, розташованих у регістрах *A* і *B*. Після виконання обчислень молодша частина добутку розміщується в акумуляторі, а старша – у регістрі-розширювачі *B*. Якщо вміст регістру *B* дорівнює 0, то ознака *OV*, скидається або встановлюється. Ознака переносу *C* завжди скидається. Наприклад, якщо на початку обчислення добутку акумулятор містить число 200 (0*C8h*), а регістр *B* – 160 (0*A*0*h*), то в результаті виконання команди `MUL AB` отримаємо добуток 32000 (7*D*00*h*). При цьому акумулятор буде містити нуль, регістр-розширювач – значення 7*DH*, ознака *OV* буде встановлена, а ознака *C* – скинута.

```
; програма
        MOV   A,0        ; скидання акумулятора
LOOP:   ADD   A,@R0      ; завантаження множеного
        MOV   B,173      ; завантаження множника
        MUL   AB         ; множення
        MOV   @R0,A      ; запис молодшого байту
                         ; часткового добутку
        INC   R0         ; приріст адреси
        MOV   A,B        ; пересилання старшого байту
                         ; часткового добутку в акумулятор
        XCH   A,@R0      ; попереднє формування чергового
                         ; байту добутку
        DJNZ  R1,LOOP    ; цикл, якщо не всі байти вихідного
                         ; числа помножені на константу
```

Отриманий добуток розміщується на місці вихідного числа і займає в РПД на один байт більше. Час обчислення складає (1+13*N*)мкс, де *N* – формат вихідного числа в байтах.

Приклад 4.18. Розробити програму, перетворення двійкового числа в *BCD*-код. Під час перетворення отримати трирозрядне *BCD*-число, старшу цифра якого (кількість сотень) розмістити у регістрі *R0*, а дві молодші – в акумуляторі. Вихідне число розташоване в акумуляторі.

Для швидкого перетворення двійкових чисел в десяткові двійково-кодовані (*BCD*-числа) може бути використана команда ділення. Команда `DIV` виконує ділення вмісту акумулятора на вміст регістру-розширювача *B*. Після виконання ділення акумулятор містить цілу частину частки, а розширювач – залишок. Прапори *C* і *OV* скидаються. Під час ділення на нуль встановлюється ознака переповнення, а частка залишається невизначеною. Час перетворення складає 16 мкс.

```
; програма
MOV     B,100 ; (B) ← 100, для обчислення кількості сотень у ;
              заданому числі
DIV     AB    ; акумулятор містить кількість сотень, тобто
              ; старшу цифру
```

```
MOV     R0,A    ; пересилання в R0 старшої цифри
XCH     A,B     ; пересилання залишку вихідного числа в
                ; акумулятор
MOV     B,10    ; (B) ← 10, для обчислення кількості десятків в
                ; числі
DIV     AB      ; акумулятор містить кількість десятків, регістр
                ; B – кількість одиниць
SWAP    A       ; розміщення кількості десятків в старшій
                ; тетраді акумулятора
ADD     A,B     ; підсумок залишку (кількості одиниць)
                ; акумулятор містить дві молодші цифри
```

4.3.7. Виконання логічних операцій

Ці команди виконують операції І, АБО, ВИКЛЮЧАЮЧЕ АБО і записують результат в байт призначення. Є команди циклічного і арифметичного зсуву вправо і вліво.

Приклад 4.19: Обчислити логічну функцію трьох змінних $Y = X \cdot \overline{V} \vee W \cdot (X \vee V)$. Змінні *X, V* і *W* надходять на виводи *P*1[2], *P*1[1] і *P*1[0] порту *P*1 відповідно. Результат *Y* необхідно видати на вивід *P*1[3] порту *P*1 у послідовному коді.

```
; програма
Y    BIT    P1.3    ; специфікація бітів порту P1
X    BIT    P1.2    ; специфікація бітів порту P1
V    BIT    P1.1    ; специфікація бітів порту P1
W    BIT    P1.0    ; специфікація бітів порту P1
     MOV    C,X     ; ввід X
     ANL    C,/V    ; виконання X AND NOT V
     MOV    F0,C    ; збереження результату в F0
     MOV    C,X     ; ввід X
     ORL    C,V     ; виконання X OR V
     ANL    C,W     ; виконання W AND (X OR V)
     ORL    C,F0    ; виконання W AND (X OR V) OR (X AND NOT V)
     MOV    Y,C     ; вивід результату
```

Ознака *F*0 використовується для тимчасового зберігання першої кон'юнкції $(X \cdot \overline{V})$. Час виконання програми складає 14 мкс.

***Приклад 4.20*:** Обчислити логічну функцію $F = 2(X1 \vee X2) + K$, де $K = X3 \& X4$, якщо $\tilde{N} = 1$, та $K = X5 - 1$, якщо $\tilde{N} = 0$. Вихідні дані розміщуються в регістрах *R*1, *R*2, *R*3, *R*4, *R*5 відповідно. Результат розмістити у комірці пам'яті за адресою 2*Ch*.

```
; програма
        MOV   A,R1      ; пересилання (A) ← (R1)
        ORL   A,R2      ; виконання (X1 ∨ X2)
        CLR   C         ; встановлення C: = 0
        RLC   A         ; зсув вліво 2(X1 ∨ X2)
        MOV   R2,A      ; пересилання результату (R2) ← (A)
        JC    Label1    ; перехід, якщо C: = 1
; обчислення виразу за умови C: = 0
        DEC   R5        ; виконання K := X5 – 1
        MOV   A,R5      ; пересилання (A) ← (R5)
        ADDC  A,R2      ; виконання 2(X1 ∨ X2) + K
        JMP   Label2    ; перехід на кінець
; обчислення виразу за умови C: = 1
Label1  MOV   A,R3      ; пересилання (A) ← (R3)
        ANL   A,R4      ; виконання K = (X3 & X4)
        ADDC  A,R2      ; виконання 2(X1 ∨ X2) + K
Label2  MOV   2Ch,A     ; розміщення результату
```

4.3.8. Виконання операцій з бітами

Працюють з бітом переносу і будь-якими іншими бітами, що допускають пряму адресацію. Наприклад, якщо *C* = 0, розряд порту *P*1.0 = 0, то виконання команди `ORL C,/P1.0` приведе до встановлення біта *C*.

***Приклад 4.21*:** Вибрати нульовий регістровий банк.

```
; програма
ANL   D0h,11100111B     ; скидання бітів RS1 і RS0
```

***Приклад 4.22*:** Встановити в одиниці біти $P1[3..0]$ порту $P1$.

```
; програма
ORL   P1,00001111B     ; P1[3..0]←1111
```

***Приклад 4.23*:** Скинути біти $P2[6]$, $P2[2,0]$ порту $P2$.

```
; програма
ANL  P2,10111010B  ; скидання бітів P2[6], P2[2,0] порту P2.
```

***Приклад 4.24*:** Інвертувати біти порту $P1$, відповідно одиничним бітам акумулятора:

```
XRL   P1,A          ; ВИКЛЮЧНЕ АБО порту P1 і акумулятора
```

***Приклад 4.25*:** Інвертувати біти $P0[7..5]$ порту $P0$.

```
; програма
XRL   A,0FH         ; ВИКЛЮЧНЕ АБО акумулятора і константи
```

***Приклад 4.26*:** Інвертувати біти $A[3..0]$ акумулятора.

```
; програма
XRL    P0,11110000B ; ВИКЛЮЧНЕ АБО порту 0 і константи
```

Приклад 4.27. Розробити програму передавання масиву розпакованих десяткових цифр, що у вихідному стані знаходиться в РПД, у зовнішній пристрій.

Управління групою бітів порту для рішення задачі відбувається за наступним протоколом. Для передачі чотирьох бітів даних використовуємо молодші виводи порту $P1$ – $P1[3..0]$. Виводи $P1[4]$ і $P1[5]$ використовуються в якості сигналів квитирування. Передавання даних на вихід МК51 супроводжує строб сигналу на виводі $P1[4]$. Зовнішній пристрій, прийнявши дані, встановлює сигнал відповіді для МК51 на вході $P1[5]$. Біти $P1[6]$, $P1[7]$ в процесі передачі даних не змінюють свого значення. Вихідними параметрами для програми є початкова адреса масиву, збережена в регістрі $R0$, і довжина масиву, збережена в регістрі $R1$. Біти порту $P1$, що не використовуються, зберігаються в незмінному вигляді.

```
; програма
        ORL     P1,00100000B    ; налагодження виводу P1[5]
                                ; на ввід
LOOP:   MOV     A,@R0           ; завантаження чергового
                                ; байта в акумулятор
        ANL     P1,11100000B    ; скидання даних і стробу
        ORL     P1,A            ; видача даних
        ORL     P1,00010000B    ; видача стробу
WAIT:   JNB     P1.5,WAIT       ; очікування відповіді
        INC     R0              ; інкремент покажчика
                                ; адреси
        DJNZ    R1,LOOP         ; повернення на початок
                                ; циклу, якщо не всі дані
                                ; передані
```

***Приклад 4.28*:** Розробити програму послідовної передачі даних з акумулятора на вивід *P*2[0] порту *P*2. Дані передати в *манчестерському коді*, коли кожний біт передається двома сигналами: перший містить інверсію біта, другий – пряме значення:

```
; програма
        MOV     R0,8        ; лічильник біт
LOOP:   RRC     A           ; встановлення ознаки C
        CPL     C           ; інверсія біта
        MOV     P2.0,C      ; передача інверсії біта
        CPL     C           ; відновлення прямого значення
                            ; біта
        NOP                 ; три команди NOP для
                            ; вирівнювання
        NOP                 ; тривалості інтервалів
        NOP
        MOV     P2.0,C      ; передача прямого значення
                            ; біта
        DJNZ    R0,LOOP     ; цикл, якщо лічильник біт
                            ; не дорівнює нулю
```

Передача виконується молодшими бітами наперед. Тривалість одного інтервалу дорівнює шести машинним циклам – 6 мкс, час передачі біта дорівнює 12 мкс, час передачі байта – 96 мкс, швидкість передачі 83 Кбіт/с або 10,4 Кб/с.

4.3.9. Виконання команд передачі управління

Дана група містить команди умовного і безумовного переходів, виклику і повернення з підпрограм. До команд безумовного переходу належать:

– довгий перехід `LJMP`, який дозволяє здійснювати передачу управління в межах всієї пам'яті програм (до 64 Кб). Команда містить 3 байти. Для економії пам'яті, якщо це можливо, використовують приведені нижче команди, що мають меншу довжину;

– абсолютний перехід `AJMP` для переходу в межах сторінки розміром 2048 байт, для чого в коді команди задаються 11 молодших розрядів адреси;

– короткий перехід `SJMP` в межах від – 128 до +127 байт відносно адреси команди, наступної за `SJMP`. Для такого переходу в другому байті команди задається зсув *rel* - ціле число із знаком. Такий же метод адресації використовують і всі команди умовного переходу.

Команда непрямого переходу `JMP @A+DPTR` дозволяє виконати перехід за адресою, невідомою наперед і обчислюваною програмою.

Команди умовних переходів дозволяють виконувати розгалуження за наступними умовами: вміст *A* рівний нулю (`JZ`), не рівно нулю (`JNZ`); перенесення, рівне нулю (`JNC`), не рівний нулю (`JC`); біт, що адресується, рівний нулю (`JNB`), не рівний нулю (`JB`, `JBC`).

Команда `DJNZ` виконує декремент вказаної комірки і здійснює розгалуження, якщо результат не рівний нулю. Інакше виконується перехід до наступної команди. Адреса переходу визначається підсумовуванням вмісту лічильника команд, збільшеного на 2, із зсувом. Цю команду зручно використовувати для організації циклів.

Команда «порівняння і перехід, якщо не рівно» – `CJNE` порівнює значення двох операндів і виконує розгалуження, якщо вони не рівні. Обидва операнди є цілими числами без знаку. Залежно

від співвідношення операндів встановлюється або скидається біт *C* (див. табл. 4.2.).

Для звернення до підпрограм використовуються команди ACALL (відносний виклик) і LCALL (довгий виклик). Ці команди на відміну від команд переходу AJMP, LJMP зберігають в стеку адресу повернення в основну програму. Для повернення з підпрограми використовується команда RET, після виконання якої програма продовжує виконання з команди, наступної за LCALL або ACALL.

Для повернення з підпрограми обробки переривання використовується команда RETI. Її відмінність від команди RET в тому, що вона дозволяє переривання обслуженого рівня. Якщо при виконанні команди RETI виявлено переривання з таким же або меншим рівнем пріоритету, одна команда основної програми виконується до обробки такого переривання. Цю властивість системи переривань можна використовувати для організації покрокового виконання програми, зручного під час налагодження контролерів.

Команди передачі управління вже були використані у розглянутих вище прикладах: умовні та безумовні переходи на мітку – у прикладі 4.20; звернення до підпрограм та повернення із підпрограм – у прикладах 4.14, 4.15; організація циклів – у прикладах 4.16, 4.17, 4.27, 4.28.

***Приклад 4.29*:** Задати таблицю переходів на різні частини програми, розпочинаючи з адреси 30*h*. Виконати перехід за параметрами:

```
; таблиця переходів (3 байта на команду)
        ORG    30h
        LJMP   Reset1       ; N = 0
        LJMP   LoadMem      ; N = 1
        LJMP   GetMem       ; N = 2
        LJMP   Go           ; N = 3
        LJMP   ContAddr     ; N = 4
        LJMP   Step         ; N = 5
```

Переходи організовані наступним чином. Параметр переходу ($N = 0 – 5$) передається з порту $P1$ і розташовується в розрядах $P1[6..4]$. За значенням параметра відбувається відповідний перехід,

наприклад, якщо $N = 0$, відбувається перехід на частину програми за параметром *Reset*1. Описані дії реалізує наступна програма.

```
; програма
      MOV   A,P1           ; читання параметру
      ANL   A,#01110000b   ; встановлення в нуль розрядів
                           ; A[7], A[3..0]
      SWAP  A              ; N в розрядах A[3..0]
      MOV   B,#3           ; занесення в регістр B константи
      MUL   AB             ; виконання (A) ← N × 3
      MOV   DPTR,#30h      ; базова адреса таблиці переходів
      JMP   @A+DPTR        ; перехід за параметром
```

4.3.10. Програмування послідовного порту

Обмін даними між двома контролерами проводиться по чотирьох лініях (рис.4.15). Виходи *TXD, RXD* - відповідно вихід передавача і вхід приймача. Сигнали *RTS* (вихід порту *P*1.1) і *CTS* (вихід порту *P*1.6) - відповідно вихід і вхід дозволи передачі.

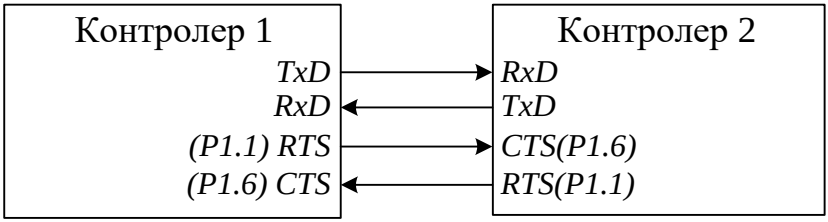


Рис. 4.15. Зв'язок між контролерами за послідовним інтерфейсом

У початковому положенні на виходах *RTS* обох контролерів встановлений рівень логічної «1» (прийом заборонений). Прийом даних (скидання *RTS*) дозволяється підпрограмою *RECEA*. Тут після установки біта *RI* (закінчення прийому байта) передача знов забороняється і отриманий байт пересилається в акумулятор. Символ «$» є спеціальним символом адреси поточної команди

```
RQTS   EQU    P1.1
; підпрограма прийому байта в ACC по RS232
ReceA: CLR    RQTS          ; дозволити передачу
       JNB    RI,$          ; чекати закінчення сеансу
                            ; прийому
```

```
        SETB   RQTS          ; заборонити передачу
        CLR    RI            ; очистити прапор закінчення
                             ; сеансу прийому
        MOV    A,SBUF        ; переслати прийнятий байт в ACC
        RET                  ; повернення з підпрограми
```

У підпрограмі передачі байта *TranA* при отриманні дозволу байт записується в *SBUF*, потім виконується очікування закінчення передачі, і очищається біт *TI*.

```
CTS     EQU   P1.6     ;
; підпрограма передачі байта з ACC за RS232
TranA: JB     CTS,$    ; чекати дозволу передачі
       MOV    SBUF,A   ; почати передачу
       JNB    TI,$     ; чекати закінчення сеансу передачі
       CLR    TI       ; очистити флаг закінчення сеансу
                       ; передачі
       RET             ; повернення з підпрограми
```

Хай частота задаючого генератора складає $f_Ч$ = 11.5 МГц. Необхідно отримати швидкість передачі за послідовним інтерфейсом, що дорівнює 9600 біт/с. УАПП працює в режимі 1, при якому формат посилки – 10 біт, а швидкість передачі управляється таймером Т/Л1 в режимі 2. З виразу (4.1) визначимо значення біта *SMOD* і значення, що завантажується в старший байт таймера *TH*1, за яких швидкість роботи послідовного порту *F* буде наближатися до 9600 біт/с. Отримаємо, що за *SMOD* = 1 і *TH*1=250, частота обміну *F* = 9982 біт/с. Відносна похибка швидкості дорівнює приблизно 4%, що є прийнятним, оскільки максимально припустима похибка складає 5%.

Хай контролер 1 отримує від контролера 2 байт лічильника, а потім байти даних, число яких передане в лічильник. Якщо лічильник рівний нулю, програма закінчується. Контролер 1 модифікує отриманий байт шляхом операції АБО з байтом із зовнішньої пам'яті даних (ці байти послідовно розташовуються в комірках ЗПД, починаючи з адреси *100h*) і передає результат контролеру 2. Програма ініціалізації і обміну даними контролера 1 може мати вигляд:

```
        MOV    IE,#0           ; заборона переривань
        MOV    TMOD,#0010000b  ; Т/Л1: режим 2
```

```
      MOV   TH1,#0FAh       ; TH1=250d для швидкості
                            ; 9600 біт/с
      MOV   TL1,TH1         ; ініціалізація TL1
      MOV   PCON,#1000000b  ; SMOD = 1
      MOV   SCON,#0101000b  ; УАПП-режим 1, дозвіл
                            ; прийому
      SETB  TR1             ; пуск Т/Л1
      ACALL ReceA           ; прийняти байт лічильника
      JZ    PEnd            ; кінець, якщо лічильник = 0
      MOV   R3,A
      MOV   RPTR,#100h
LdM:  ACALL ReceA           ; прийняти байт даних
      MOV   B,A             ; зберегти байт
      MOVX  A,@DPTR         ; читання байта з ЗПД
      INC   DPTR
      ORL   A,B             ; модифікація байта
      ACALL TranA           ; передача байта
      DJNZ  R3,LdM          ; повернення на початок циклу
PEnd: NOP
      END
```

4.3.11. Арифметичні операції з плаваючою комою

Розглядається реалізація операцій з плаваючою комою, в яких числа подані у форматі, прийнятому фірмою *INTEL*. Число M з плаваючою комою подається у вигляді

$$M = m \times 2^{Pm}$$

де m - нормалізована мантиса, тобто мантиса представлена в діапазоні від 0,5 до 1; Pm – порядок.

Під час запису нормалізованого числа в машинному вигляді старший розряд мантиси m, за визначенням рівний одиниці, опускається. Такий формат розширює поле запису порядку на один розряд. Машинний порядок числа P подається в зміщеній формі і записується у вигляді $P = Pm + 126$. Максимальне значення P складає 254, мінімальне – 1. Значення P, що дорівнюють 255 і 0 використовуються для подання спеціальних чисел. Число нуль записується нульовим порядком і нульовою мантисою.

Формат чисел із плаваючою комою зображений на рис. 4.16.

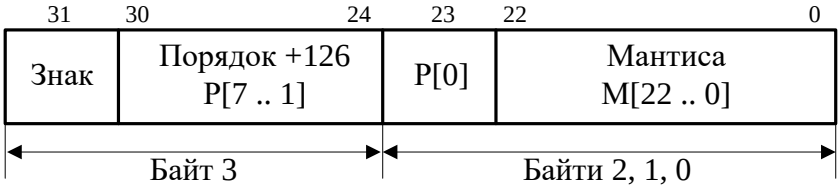


Рис. 4.16. Формат подання чисел з плаваючою комою

***Приклад 4.30*:** Розробити програму множення чисел *A* і *B*, поданих у форматі з плаваючою комою.

Під час множення чисел *A* і *B*, поданих у форматі з плаваючою комою, мантиса результату рівна добутку мантис операндів, а порядок – сумі порядків операндів:

$$y = A \cdot B = (a \cdot 2^{Pa})(b \cdot 2^{Pb}) = (a \cdot b) \cdot 2^{(Pa+Pb)}$$

Множення трибайтних мантис здійснюється за допомогою команди `MUL`, що входить до системи команд МК.51.

Команда `MUL` виконує множення однобайтних чисел з отриманням двобайтного результату. Остаточний результат формується шляхом підсумовуванням часткових добутків з урахуванням позиції кожного з них. Матриця часткових добутків при множенні мантис приведена на рис.4.17, де a_i, b_i - байти множеного і множника, причому a_2 і b_2 старші, а a_0 і b_0 - молодші байти. Звідси видно, що

$$a \cdot b = 2^{32} \cdot a_2 \cdot b_2 + 2^{24}(a_2 \cdot b_1 + a_1 \cdot b_2) + \\ + 2^{16}(a_2 \cdot b_0 + a_1 \cdot b_1 + a_0 \cdot b_2) + 2^{8}(a_1 \cdot b_0 + a_0 \cdot b_1) + a_0 \cdot b_0$$

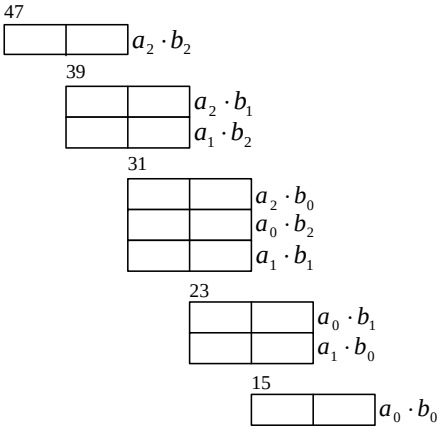


Рис. 4.17. Матриця часткових добутків під час множення мантис

Для отримання *n*-розрядного результату множення операційний пристрій повинен містити $(n + \log_2 n)$ розрядів, що в даному випадку складає 30 розрядів. Тоді, як видно з рис. 4.17, немає необхідності використовувати молодші байти часткових добутків $a_1 \cdot b_0$ та $a_0 \cdot b_1$, а також часткового добутку $a_0 \cdot b_0$, що дозволяє прискорити отримання результату. Результат множення мантис може вийти ненормалізованим. В цьому випадку необхідно нормалізувати мантису добутку з відповідною корекцією порядку.

Множення виконується за наступним алгоритмом:

1. Аналіз операндів на рівність нулю і привласнення результату нульового значення, якщо один з операндів рівний нулю.
2. Обробка порядків і визначення знаку результату.
3. Множення мантис.
4. Округлення мантиси; та за необхідності виконання її нормалізації з корекцією порядку.

В пункті 4 виконується перевірка виходу результату за межі розрядної сітки – антипереповнювання (добуток менше мінімального для даного формату числа) або переповнювання (добуток більше максимального) і за необхідності встановлюються відповідні ознаки.

Обробка порядків і визначення виходу результату за межі розрядної сітки. Рядки операндів перетворяться із зміщеної форми шляхом віднімання константи 126 і представляються в додатковому коді у вигляді двобайтного числа, причому в старшому байті -

розмножений знаковий розряд. Перетворені порядки двох операндів складаються. Операнд *A* розташовується в комірках з адресами 20*h* – 23*h*, а операнд *B* – в комірках 24*h*–27*h* резидентної пам'яті даних, причому старші байти розташовуються в старших адресах (наприклад, порядок – в байті з адресою 23*h* і таке інше).

Для звернення у програмі до кожного окремого байту чотирирозрядних операндів застосовувались наступні символічні позначення: операнду *X* відповідають комірки РПД *OPXp* – *OPX*0, операнду *B* – комірки *OPYp* – *OPY*0. Де, *OPXp* – байт порядку, *OPX*2, *OPX*1, *OPX*0 – відповідно другий, перший та нульовий байти мантиси.

```
; формування порядку результату
OpX0    EQU  20h          ; молодший байт мантиси A
OpX1    EQU  21h          ; середній байт мантиси A
OpX2    EQU  22h          ; старший байт мантиси A
OpXp    EQU  23h          ; порядок A
OpY0    EQU  24h          ; молодший байт мантиси B
OpY1    EQU  25h          ; середній байт мантиси B
OpY2    EQU  26h          ; старший байт мантиси B
OpYp    EQU  27h          ; порядок B и Y
Sg      EQU  2Ah          ; ознаки
SgX     EQU  2Bh          ; знаковий байт порядку A и Y
SgY     EQU  2Ch          ; знаковий байт порядку B
;обробка порядків
        MOV  SgX,#0
        MOV  SgY,#0
;формування порядку B
        MOV  A,OpYp
        MOV  C,OpY2.7     ; молодший розряд порядку
        RLC  A
        CLR  C
        SUBB A,#126       ; віднімання константи
        MOV  OpYp,A       ; збереження порядку B
        JNB  C,Xx1        ; формування знакового байта
        MOV  SgY,#0FFh    ; порядку B
;формування. порядку A
Xx1:    MOV  A,OpXp
        MOV  C,OpX2.7     ; молодший розряд порядку
        RLC  A
```

```
        CLR   C
        SUBB  A,#126        ; порядок A
        JNB   C,Xx2         ; формування знакового байта
        MOV   SgX,#0FFh     ; порядку A
Xx2:    ADD   A,OpYp        ; додавання порядків
        MOV   OpYp,A        ; збереження Рмол
        MOV   A,SgX         ;
        ADDC  A,SgY         ;
        MOV   SgY,A         ; збереження Рст
```

В результаті виконання програми отримуємо двобайтний перетворений порядок результату *P*пр (у доповнювальному коді). Він складається із старшого *P*ст і молодшого *P*мол байтів. Далі здійснюється множення мантис та округлення порядку за потреби.

Програма множення трибайтних мантис. Добуток накопичується в регістрах *R*5 (старший), *R*4, *R*3 *і R*2. У програмі використовується підпрограма множення *Mul3*, яка призначена для множення і підсумовування часткових творів $a_2 \cdot b_0$, $a_0 \cdot b_2$, $a_1 \cdot b_1$

```
;продовження програми
;підпрограма множення
Mul3:  MUL     AB
       ADD     A,R2
       MOV     R2,A
       MOV     A,B
       ADDC    A,R3
       MOV     R3,A
       CLR     A
       ADDC    A,R4
       MOV     R4,A
       RET
; множення мантис
       SETB    OpY2.7      ; встановлення старших
       SETB    OpX2.7      ; розрядів мантис
       MOV     A,OpX1      ; формування a1·b0
       MOV     B,OpY0
       MUL     AB
       MOV     R2,B
       MOV     A,OpX0      ; формування a0·b1
```

```
MOV     B,OpY1
MUL     AB
MOV     A,B          ; формування старшого байта
ADD     A,R2         ; y1 = a0 · b1 + a1 · b0
MOV     R2,A
CLR     A
ADDC    A,#0
MOV     R3,A
MOV     A,OpX2       ; a2 · b0
MOV     B,OpY0
ACALL   Mul3         ; y2 = a2 · b0 + y1
MOV     A,OpX1       ; формування a1 · b1
MOV     B,OpY1
ACALL   Mul3         ; y3 = a1 · b1 + y2
MOV     A,OpX0       ; формування a0 · b2
MOV     B,OpY2
ACALL   Mul3         ; y4 = a0 · b2 + y3
MOV     A,OpX1       ; формування a1 · b2
MOV     B,OpY2
MUL     AB
ADD     A,R3         ; y5 = y4 + a1 · b2 молодший байт
MOV     R3,A
MOV     A,B
ADDC    A,R4         ; y5 = y4 + a1 · b2 старший байт
MOV     R4,A
CLR     A
ADDC    A,#0
MOV     R5,A
MOV     A,OpX2       ; формування a2 · b1
MOV     B,OpY1
MUL     AB
ADD     A,R3         ; y6 = y5 + a2 · b1 молодший байт
MOV     R3,A
MOV     A,B
ADDC    A,R4         ; y6 = y5 + a2 · b1 старший байт
```

```
MOV     R4,A
CLR     A
ADDC    A,R5
MOV     R5,A
MOV     A,OpX2
MOV     B,OpY2      ; a2 · b2
MUL     AB
ADD     A,R4        ; y7 = y6 + a2 · b2 молодший байт
MOV     R4,A
MOV     A,B
ADDC    A,R5        ; y7 = y6 + a2 · b2 старший байт
MOV     R5,A
```

Після множення мантис необхідно виконати аналіз порядку результату на переповнювання і антипереповнювання.

Алгоритм визначення виходу за межі розрядної сітки приведений на рис.4.18. Спочатку визначається, чи було переповнювання при додаванні молодших байтів мантис операндів. Це має місце, якщо Pc=254*d*=11111110*b* і визначає антипереповнювання результату, а також, якщо Pc=1 (переповнювання). Якщо переносу з молодшого байта в старший під час додавання не відбулося, необхідно проаналізувати значення молодшого байту порядку результату. Максимальне значення зміщеного порядку рівне 254, що в перетвореному порядку складає 254–126=128=80*h*. Тому, якщо молодший байт позитивного порядку більше 80*h*, фіксується переповнювання.

Мінімальне значення зміщеного порядку дорівнює 1, що в перетвореному порядку складає

$$\begin{array}{r} 0.00000001 \\ -\ 0.01111110 \\ \hline 1.10000011 \end{array}$$

Тому, якщо молодший байт негативного порядку менше 83*h*, фіксується антипереповнювання.

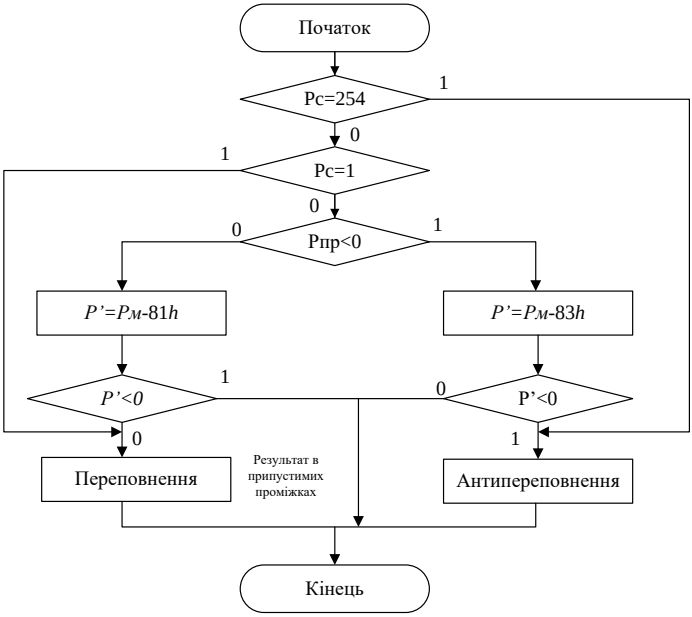


Рис. 4.18. Алгоритм визначення переповнення і антипереповнювання

Алгоритм на рис. 4.18 реалізуємо за допомогою наступної програми:

```
; продовження програми
; визначення переповнення і антипереповнювання
       JNB     P,Xx5          ; аналіз на парність старшого байта
                              ; порядку, було перенесення
                              ; в старший байт порядку
        JNB    SgY.7,Xx4      ;
Xx20:  SETB    Sg.4           ; встановлення ознаки
                              антипереповнювання
        AJMP   Xx9            ;перехід на кінець
Xx4:   SETB    Sg.5           ; встановлення ознаки
                              переповнювання
                   ;не було перенесення в старший байт порядку
Xx5:  MOV      A,OpYp         ; молодший байт порядку
      CLR      C              ;
      JNB      SgY.7,Xx6      ;
      SUBB     A,#83h         ; порядок від'ємний
```

```
      JB      Cy,Xx20     ; антипереповнювання
      AJMP    Xx9         ; перехід на кінець
Xx6:  SUBB    A,#81h      ; порядок додатний
      JNB     Cy,Xx4      ; переповнювання
Xx9:  NOP                 ; закінчення обробки порядків
```

Програма множення чисел з плаваючою комою, частини якої були приведені вище, вимагає 275 байт пам'яті програм. Час множення чисел складає 350 – 410 циклів роботи мікроконтролера, що, наприклад, при частоті кварцового резонатора, рівній 12 мгц складає 0,35-0,41 мс.

4.3.12. Обчислення складних функцій

***Приклад 4.31*:** Розробити програму пересилання масиву з двадцяти слів із порту *P*0 у резидентну пам'ять даних, розпочинаючи з адреси 52*h*. Розробити алгоритм та програму обчислення заданого виразу, якщо його аргументами є перші шість елементів масиву. Результат обчислення функції розмістити у регістрах *R*7, *R*6 нульового банку регістрів.

Вираз для обчислення:

$$F = 16 \times (X1 + X2 - 1) + (X3 - X4) + (X5 \times X6)/16 .$$

Алгоритм виконання завдання зображений на рис. 4.19.

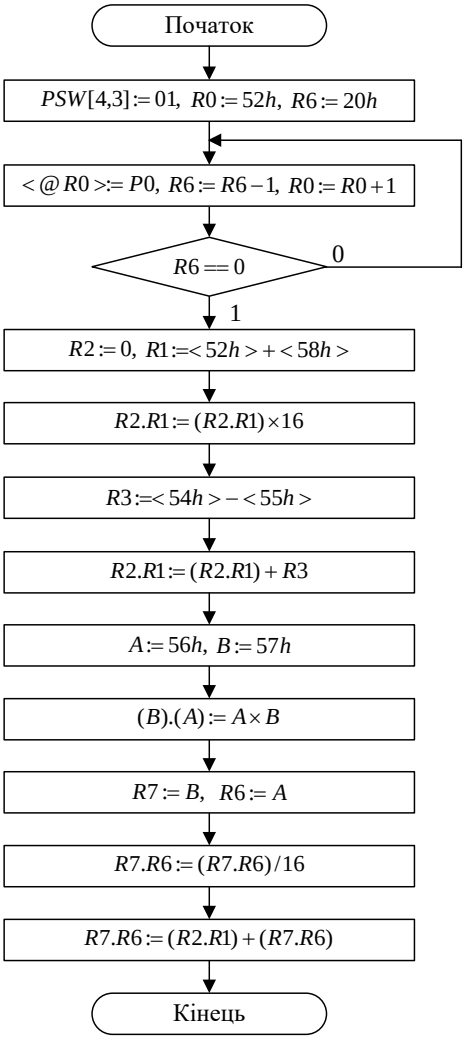


Рис. 4.19. Алгоритм обчислення виразу

```
; програма
; пересилання масиву даних
        ANL   PSW,#e7h     ; PSW[4..3]:= 00
        ORL   PSW,#08h     ; вибір першого банку регістрів
        MOV   R0,#52h      ; завантаження в регістр адреси
                           ; пам’яті
```

```
        MOV   R6,#20        ; завантаження лічильника
DDD1:   MOV   @R0,P0        ; завантаження масиву
        INC   R0            ; підготовка наступної адреси
        DJNZ  R6,DDD1       ; вихід із циклу, якщо вміст R6
                            ; дорівнює нулю
        ANL   PSW, #e7h     ; вибір нульового банку регістрів
        MOV   ACC,52h       ; завантаження акумулятора
                            ; значенням X1 = 52h:  A:=52h (X2)
        ADD   A,53h         ; A:= 53h (X2) + 52h (X1)
        DEC   A             ; декремент A:=A – 1
        MOV   R1,A          ; пересилання R1:= A
; зсув вліво вмісту A на чотири розряди
        MOV   R7,#4         ; завантаження лічильника
        MOV   R2,#0h
DDD2:   CLR   C             ; встановлення C:= 0
        MOV   A,1h          ; пересилання A:= R1
        RLC   A             ; зсув регістру R1
        MOV   R1,A
        MOV   A,R2          ; пересилання A:= R2
        RLC   A             ; зсув регістру R2
        MOV   R2,A
        DJNZ  R7,DDD2       ; вихід із циклу, якщо вміст R7 =0
        MOV   A,54h         ; завантаження A:=54h (X3)
        CLR   C             ; встановлення C:= 0
        SUBB  A,55h         ; A:= 54h (X3) – 55h (X4)
        MOV   R3,A          ; пересилання R3:= A
; визначення суми (R2.R1+R3)
        CLR   C
        ADDC  A,R1          ; підсумовування A:= R1+R3
        MOV   R1,A          ; пересилання R1:= A
        MOV   A,R2          ; пересилання A:= R2
        ADDC  A, #0         ; підсумовування A:=R2+0+C
        MOV   R2, A         ; пересилання R2:= A
        MOV   A,56h         ; завантаження A:= 56h (X5)
        MOV   B,57h         ; завантаження B:= 57h (X6)
        MUL   AB            ; множення (B. A):= B× A
        MOV   R7,F0h        ; пересилання R7:=B
                            ; адреса регістра B – F0h
        MOV   R6,A          ; пересилання R6:= A
```

```
; зсув вправо R7.R6 на чотири розряди
        MOV   R4,#4h        ; завантаження лічильника
DDD3:   CLR   C             ; встановлення C:= 0
        MOV   A,R7          ; пересилання A:= R7
        RRC   A             ; зсув регістру R7
        MOV   R7,A
        MOV   A,R6          ; пересилання A:= R6
        RRC   A             ; зсув регістру R6
        MOV   R6,A
        DJNZ  R4, DDD3      ; вихід із циклу, якщо R4 = 0
        CLR   C             ; встановлення C:= 0
        MOV   A,R6
        ADDC  A,R1          ; підсумовування A:= R6+R1
        MOV   R6,A
        MOV   A,R7
        ADDC  A,R2          ; підсумовування A:= R7+R2+C
        MOV   R7,A
        END
```

***Приклад 4.32*:** Розробити алгоритм обчислення $Z = \sqrt{X}$, числа з плаваючою комою.

Квадратний корінь з числа з плаваючою комою обчислюється за таким способом:

$$Z = \sqrt{M_X \cdot 2^{P_X}} = \begin{cases} 2^{\frac{P_X}{2}} \sqrt{M_X}, & \text{якщо } Px \text{ парне;} \\ 2^{\frac{P_X+1}{2}} \sqrt{\frac{M_X}{2}}, & \text{якщо } Px \text{ непарне.} \end{cases}$$

Таким чином, для добування квадратного кореня з числа із плаваючою комою необхідно порядок числа поділити на два, а з мантиси добути квадратний корінь за правилами для чисел з фіксованою комою.

Ділення порядку на два відбувається шляхом зсуву його на один розряд вправо, якщо порядок парний. Якщо порядок непарний,

то до нього необхідно додати одиницю, та зсунути мантису на один розряд вправо.

Для виконання операції використовується алгоритм, який дозволяє найпростіше обчислити квадратний корінь в системі команд МК51. Операційна схема обчислення квадратного кореня приведена на рис. 4.20 і містить *n*-розрядні регістри *X* і *D* для зберігання операнду і результату, (*n*+2)-розрядний суматор і регістр *B* для формування і зберігання проміжних результатів. На суматорі проводиться підсумовування вмісту регістрів *B* і *D*, а результат підсумовування записується в регістр *B*. Входи двох молодших розрядів *CM* підключені до логічної одиниці. Всі три регістри мають можливість зсуву коду вліво.

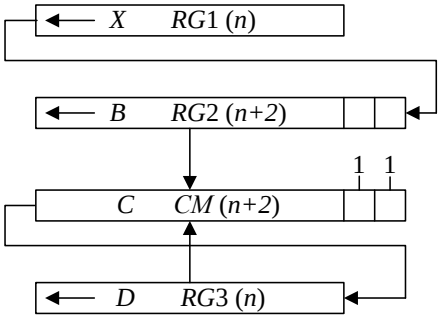


Рис. 4.20. Операційна схема обчислення квадратного кореня

У початковому стані регістри *X* і *B* скинуті, розряди регістра *D* встановлені в одиничний стан. Обчислення проводяться в *n* циклах, в кожному з яких виконуються наступні дії:

– двічі проводиться зсув вліво регістрів *X* і *B*, причому розряди, що висуваються з *X,* записуються в регістр *B*;

– виконується підсумовування (*B* + *D*); по значенню перенесення визначається значення чергового розряду результату: $Zi = (-C)$ (відмітимо, що результат формується в зворотному коді);

– у випадку, якщо $C = 0$, виконується відновлення залишку, тобто операція (*B* – *D*);

– проводиться лівий зсув регістра *D*, причому інверсне значення чергового розряду результату записується в молодший розряд регістра *D*.

У даній програмі виконується 25 циклів обчислень, причому, оскільки в останніх тринадцяти циклах цифри операнда рівні нулю,

в цих циклах зсув регістра *X* не проводиться. У цих останніх циклах виконується аналіз на нуль залишку. Якщо $B = 0$, операція закінчується. Після цього в регістрі *D* проводиться інвертування результату і його округлення, внаслідок чого формується двадцяти чотири розрядна мантиса результату.

Програма виконання двадцяти п'яти ітерацій обчислення квадратного кореня мантиси приведена нижче. Регістру *X* на рис. 4.20 відповідають комірки РПД *OPXp* – *OPX*0, регістру *B* – регістри загального призначення *R*5 – *R*2, регістру *D* – комірки *OPYp* – *OPY*0. Де *OPXp* – байт порядку у читирибайтному форматі числа із плаваючою комою, *OPX*2, *OPX*1, *OPX*0 – відповідно другий, перший та нульовий байти мантиси. Під час виконання програми використовуються підпрограми перетворення чотирибайтних чисел – зсуву вліво, додавання, віднімання і перевірки на рівність нулю.

```
; підпрограма зсуву вліво
; в R0-початкова адреса слова з чотирьох байтів
ResNorm: MOV     R7,#4
         CLRC                    ; завантаження лічильника
Rnorm:   MOV     A,@R0
         RLC     A
         MOV     @R0,A
         INC     R0
         DJNZ    R7,RNorm
         RET
; підпрограма додавання мантис <R5, R4 ,R3, R2> + @R0
AdMant1: CLR     C
         MOV     R1,#2
         MOV     R7,#4
AdMant:  MOV     A,@R1
         ADDC    A,@R0
         MOV     @R1,A
         INC     R0
         INC     R1
         DJNZ    R7,AdMant
         RET
; підпрограма віднімання мантис <R5, R4 ,R3, R2> – @R0
SuMant1: CLR     C
         MOV     R1,#2
```

```
          MOV     R7,#4
SuMant:   MOV     A,@R1
          SUBB    A,@R0
          MOV     @R1,A
          INC     R0
          INC     R1
          DJNZ    R7,SuMant
          RET
```

; підпрограма перевірки операнда на рівність нулю
; операнд знаходиться в чотирьох байтах РПД,.
; адреса старшого – в *R*0, ознака нуля – в біті *Sg*.3

```
Zero:     CLR     Sg.3
          MOV     R7,#4
Zero1:    CJNE    @R0,#0,Zend
          DEC     R0
          DJNZ    R7,Zero1
          SETB    Sg.3
Zend:     RET
OpZ1p     EQU     29h
```

; підпрограма обчислення квадратного кореня мантиси числа

```
          MOV     OpZ1p,#25     ; лічильник
Qx4:      MOV     A,OpZ1p
          CLR     C
          SUBB    A,#13
          MOV     Sg.1,C        ; Sg.1  – ознака номеру циклу
          JB      C,Qx5         ; i >13
```

; цикл для $i > 13$

```
          MOV     R0,#20h
          CLR     C
          ACALL   ResNorm       ; зсув X
          MOV     R0,#2
          ACALL   ResNorm       ; X[i] → b[0], зсув B
          MOV     R0,#20h
          CLR     C
          ACALL   ResNorm       ; зсув X
          MOV     R0,#2
          ACALL   ResNorm       ; X[i+1] → b[0], зсув B
          AJMP    Qx6
```

```
; цикл для i ≥ 13
Qx5:    MOV     R0,#2
        CLR     C
        ACALL   ResNorm     ; зсув B на 2 розряди
        MOV     R0,#2       ; вліво
        CLR     C           ;
        ACALL   ResNorm     ;
; додавання B+D
Qx6:    MOV     R0,#24h
        ACALL   AdMant1
        CPL     C
        MOV     Sg.6,C      ; збереження Zi
        JNB     C,Qx7       ; залишок додатній
; відновлення залишку
        MOV     R0,#24h
        ACALL   SuMant1
Qx7:    MOV     R0,#24h
        SETB    C
        ACALL   ResNorm     ; зсув D вліво
        MOV     C,Sg.6      ; Zi → d2
        MOV     OpY0.2,C
        JNB     Sg.1,Qx10
; якщо в останніх тринадцяти циклах залишок дорівнює нулю,
; кінець операції
        MOV     R0,#5
        ACALL   Zero
        JB      Sg.3,Qx11   ; залишок = 0
Qx10:   DJNZ    OpZ1p,Qx4
Qx11:   NOP                 ; кінець обчислення
```

Час виконання операції складає 3000–5000 циклів роботи мікроконтролера для різних операндів

4.4. Схеми підключення мікросхем

Схеми підключення периферійних пристроїв і ЗП до МК51 приведені на рис. 4.21 –4.26.

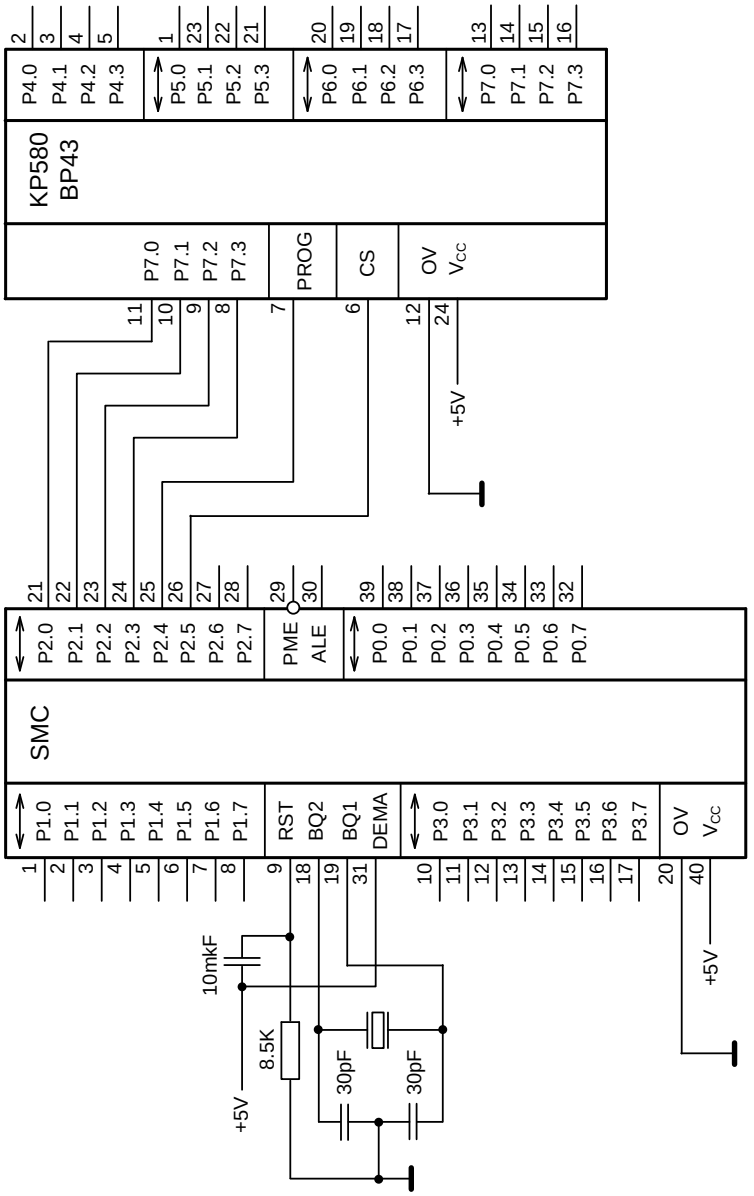


Рис.4.21. Схема розширення портів вводу/виводу з використанням мікросхеми КР580ВР43

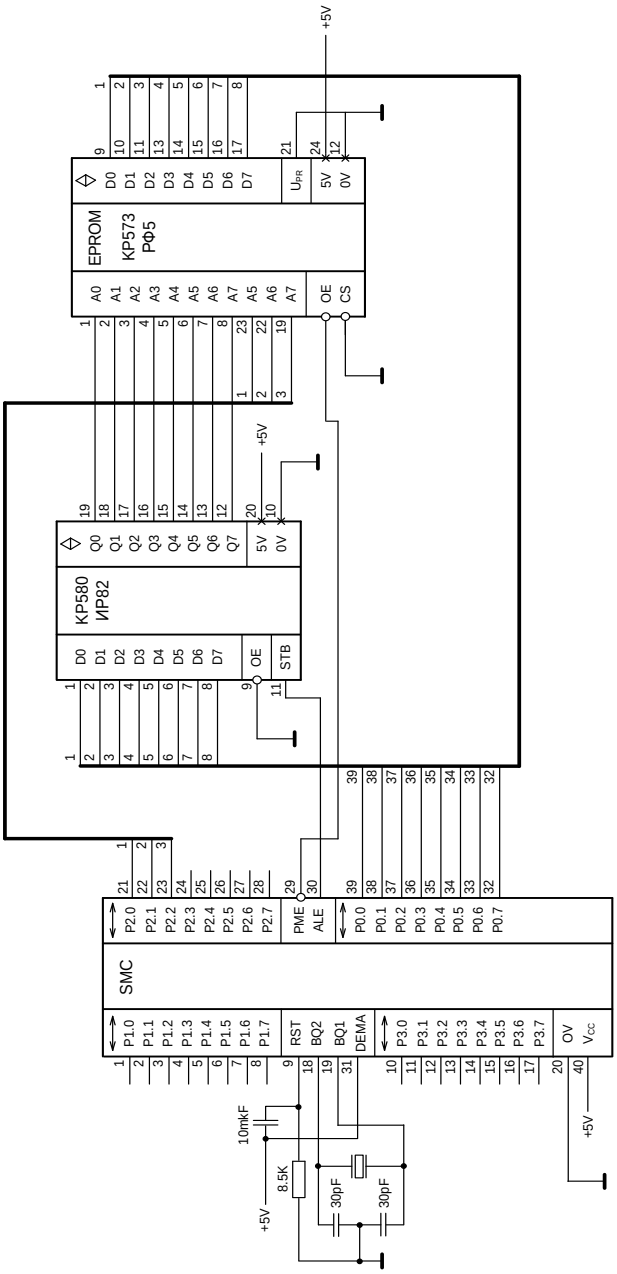


Рис.4.22. Схема підключення зовнішньої пам'яті програм з використанням МС К573РФ5

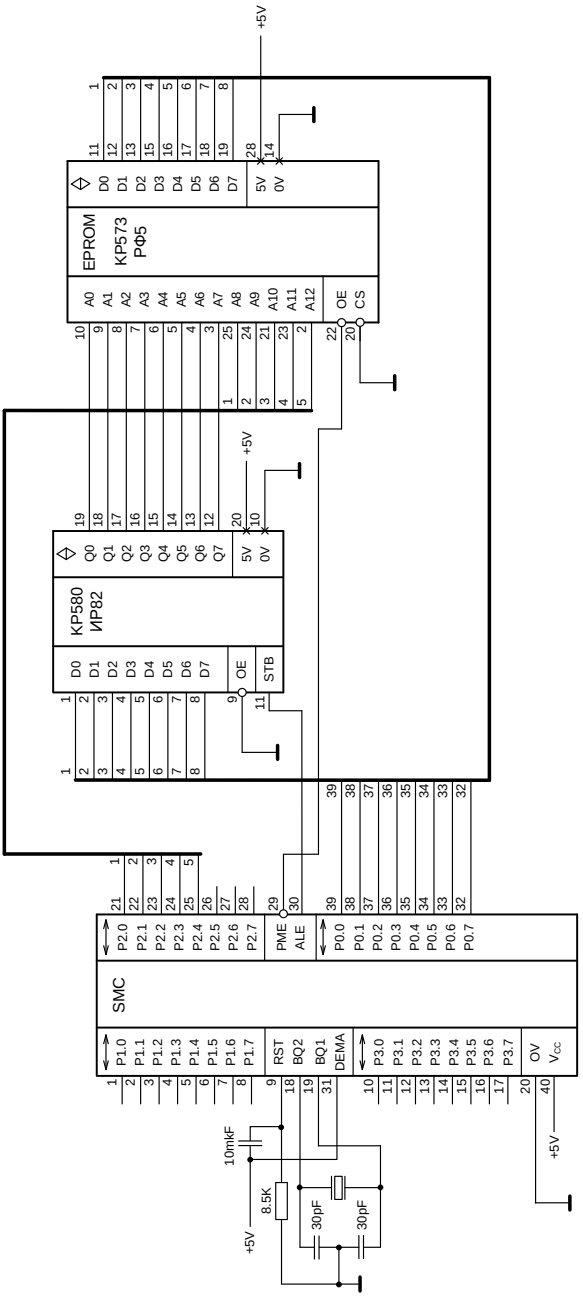
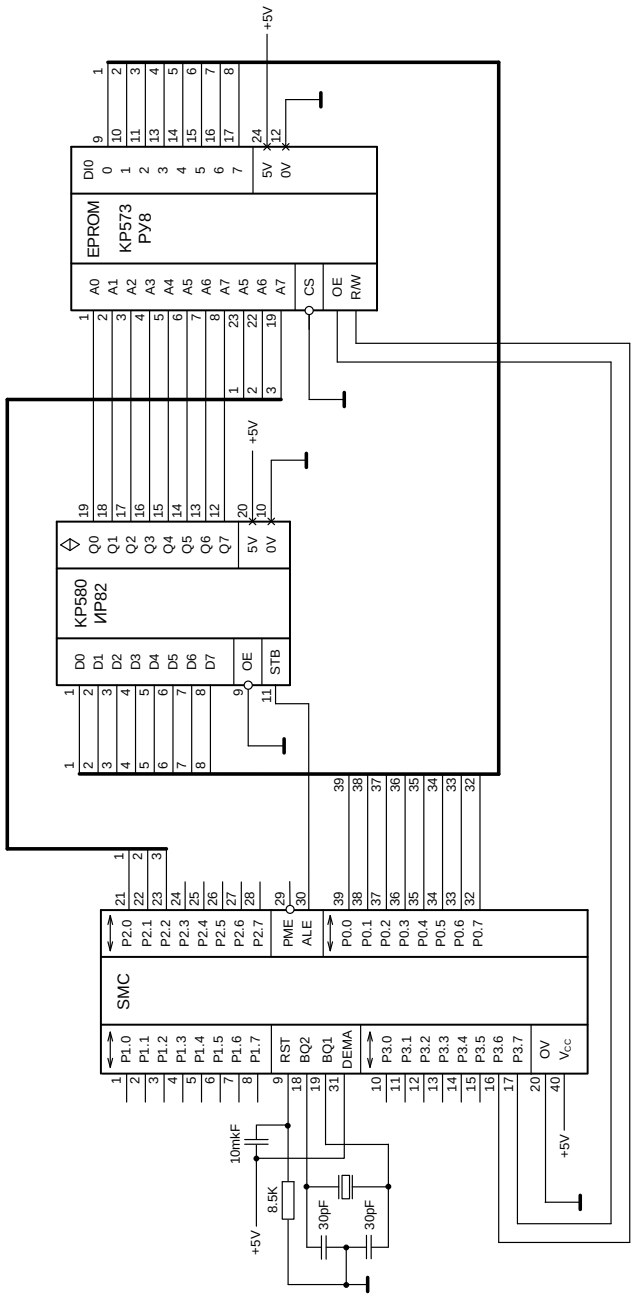
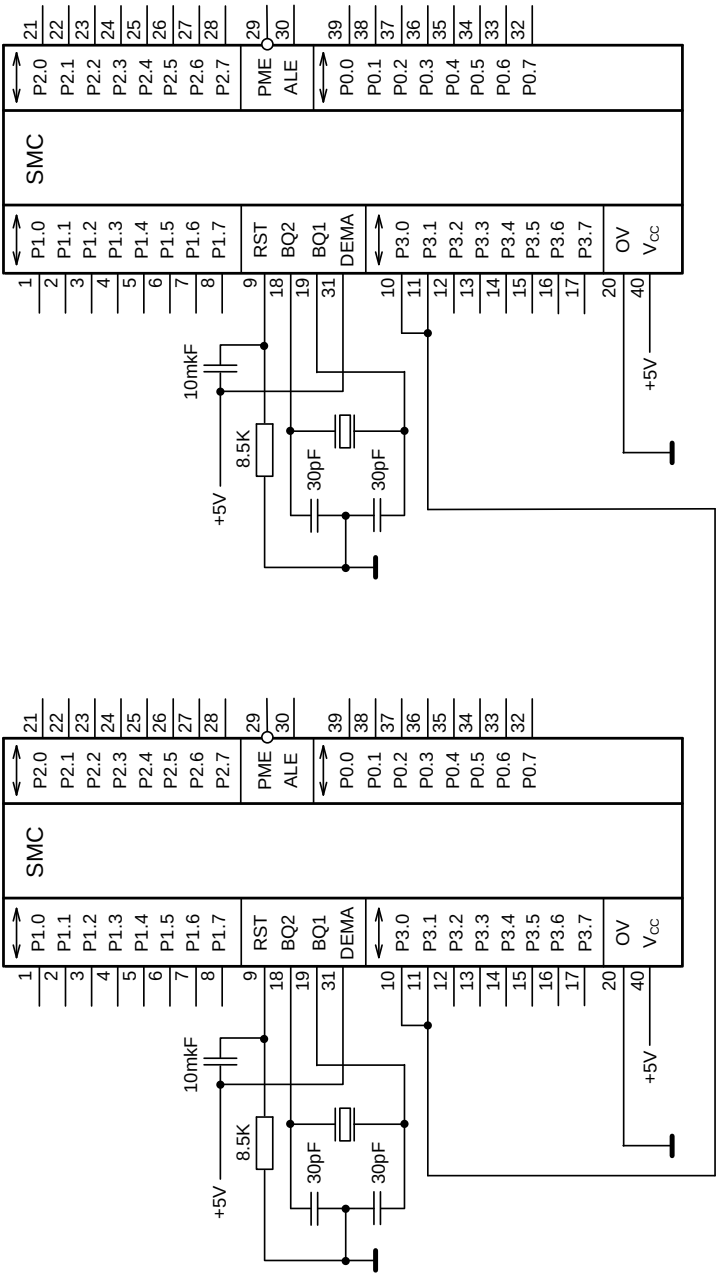


Рис.4.23. Схема підключення зовнішньої пам'яті програм з використанням ИС К573РФ6



Примітка: виводи 15, 16 мають альтернативне значення: P3.1 – $\overline{WR}$, P3.7 – $\overline{RD}$

Рис.4.24. Схема підключення зовнішньої пам'яті програм з використанням МС КР537РУ8



Примітка: Виводи 10, 11 мають альтернативне значення: P3.0 – RxD, P3.1 – TxD

Рис.4.25. Схема організації напівдуплексного послідовного каналу з використанням пари МС

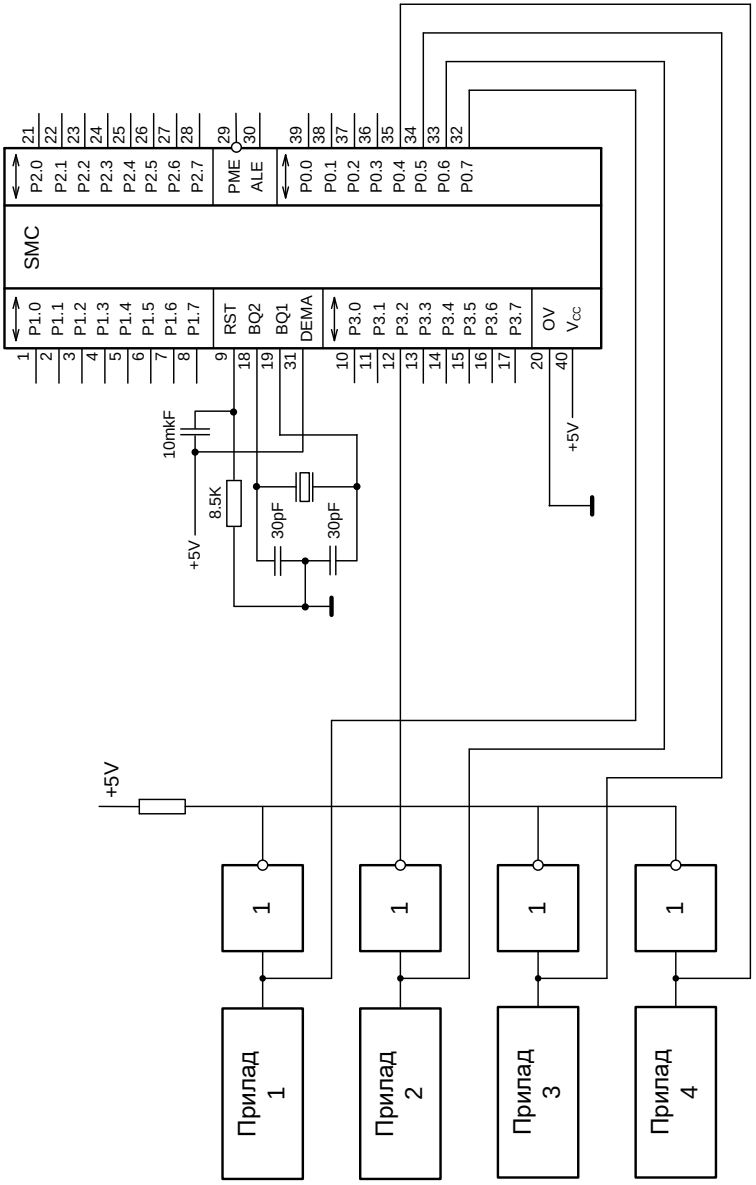


Рис. 4.26. Схема збільшення числа джерел переривання МС

5. СИСТЕМА ПЕРЕРИВАНЬ

5.1. Загальні поняття

Під перериваннями розуміють тимчасове припинення виконання програми і перехід на підпрограму з можливістю повернення на перервану програму.

Розрізняють внутрішні і зовнішні переривання. Внутрішні переривання у свою чергу поділяються на програмні і апаратні. Програмні переривання є ефективним засобам для виклику стандартних підпрограм базової системи вводу-виводу, дозволяють спростити процес налагодження програм взаємодії процесора із зовнішніми пристроями, які, можливо, розробляються паралельно з програмним забезпеченням і таке інше. Внутрішні апаратні переривання дозволяють процесору відреагувати на непередбачені ситуації (наприклад, переповнювання розрядної сітки, спрацьовування внутрішнього таймера, збій при зверненні до магістралі) і скоректувати обчислювальний процес, змінити режим роботи або, принаймні, провести безпечне зупинення системи.

Існують векторні і безвекторні зовнішні переривання. Запити на безвекторні переривання поступають на спеціальні входи процесорів. Ці запити мають більший пріоритет, ніж запити на векторні переривання. Механізм обробки безвекторних переривань закладений в процесорі на мікропрограмному або апаратному рівні. У цілому, механізм забезпечує перехід до підпрограми за визначеної адреси (із запам'ятовуванням адреси повернення і стану перерваної програми). Безвекторні переривання, як правило, використовуються процесором на його локальній магістралі, причому, входи запитів мають визначене призначення (відключення живлення, спрацьовування зовнішнього таймера і таке інше).

Зовнішні векторні переривання є важливим засобом синхронізації процесів в мікропроцесорній системі. За допомогою сигналів зовнішніх переривань пристрої системи оповіщають один одного про готовність до передачі даних або про змінення режимів роботи, тобто обмінюються управляючою інформацією. Якщо для прийому сигналів переривань в процесорах передбачені спеціальні входи, то для формування сигналів вимоги переривання можуть знадобитися додаткові засоби.

Зовнішні векторні переривання реалізуються за допомогою спеціального контролера наступним чином.

По запитах від зовнішніх пристроїв контролер переривань із урахуванням системи пріоритетів видає на процесор сигнал вимоги переривання *IRQ*. Умовою формування такого сигналу є готовність пристроїв до взаємодії з процесором і відсутність маскування запитів з боку процесора.

Отримавши сигнал *IRQ*, процесор завершує до кінця виконання чергової команди, видає на контролер сигнал підтвердження переривання *IACK* і зчитує з шини даних вектор, який виставляє на шину контролер переривання.

Процесор зберігає (зазвичай в стеку) адреси повернення і стан програми, що тимчасово перервана, обчислює адресу переходу, використовуючи для цього отриманий вектор переривання й після цього здійснює перехід на першу команду підпрограми обробки переривання. Процесор виконує підпрограму. Остання команда підпрограми є спеціальною командою повернення з переривання, яка відновлює стан перерваної основної програми і передає їй управління.

Схема підключення централізованого контролера пріоритетних переривань (КПП) показана на рис. 5.1. Зовнішні пристрої (ЗП), в числі яких можуть бути і інші процесори, формують запити на переривання *IRQ*. Особливість такої схеми полягає в тому, що підключення ЗП до магістралі процесора не є обов'язковим. Це пояснюється тим, що вектор *V* в процесор передає КПП, а не зовнішні пристрої. Завдяки цьому через переривання можуть взаємодіяти процесори, підключені до різних магістралей.

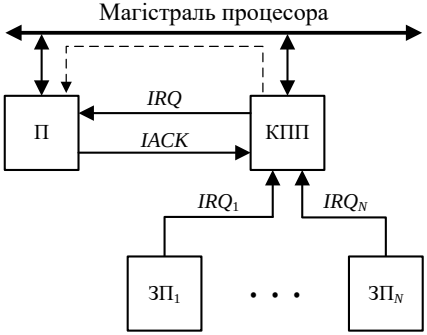


Рис. 5.1. Система с централізованим контролером пріоритетних переривань

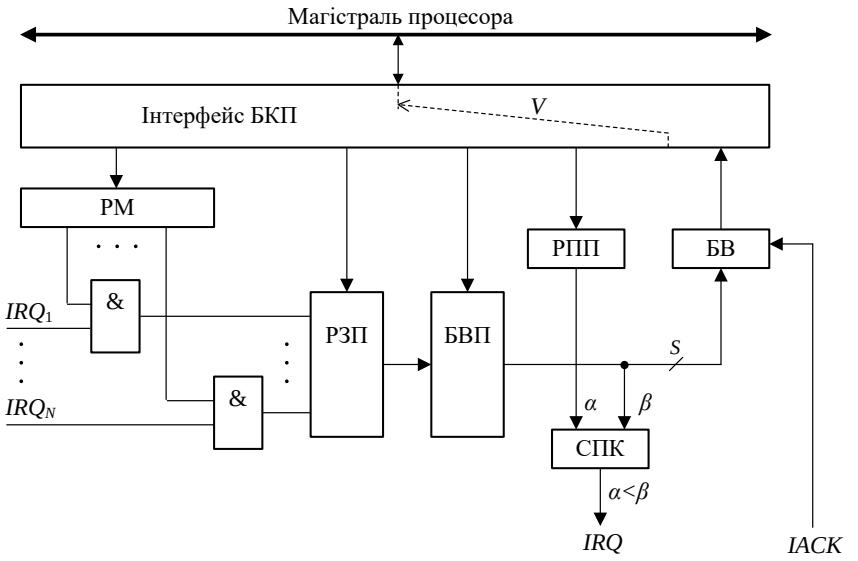


Рис. 5.2. Централізований КПП

Можливий варіант побудови централізованого КПП показаний на рис. 5.2. Контролер підключений до магістралі процесора через інтерфейс, який забезпечує процесору доступ до регістру маски (РМ) і регістру поточного пріоритету (РПП). Адреси вказаних регістрів включені в адресний простір процесора. Записом слова маски в регістр РМ процесор може дозволити або заборонити переривання від певних джерел. Незамасковані запити *IRQ*, записуються в регістр запитів переривань (РЗП). Блок вибору пріоритету (БВП) формує код самого старшого пріоритету, який в схемі порівняння кодів (СПК) порівнюється з поточним пріоритетом – пріоритетом виконуваної процесором програми. Якщо запрошений пріоритет вище поточного, то формується сигнал *IRQ*. Далі, у відповідь сигналу *IACK*, через буфер вектора (БВ) і інтерфейс в магістраль видається вектор, який приймається процесором. У регістр РПП записується новий код пріоритету. У даному контролері роль вектора виконує код номера запиту.

До достоїнств централізованих контролерів слід віднести наступне:

– можливість динамічно змінювати стратегію обслуговування заявок;

– швидке вибіркове маскування запитів на переривання.

Централізований КПП потенційно дозволяє забезпечувати різні дисципліни обслуговування заявок, оскільки всі заявки поступають в один пристрій. Для забезпечення пріоритетного обслуговування застосовується блок вибору пріоритету. У простому випадку в якості БВП використовується пріоритетний шифратор (ПШ), який формує код старшого рівня пріоритету. При цьому всі запити мають фіксовані пріоритети, рівень яких визначається номером входу КПП. За рахунок ускладнення схеми можна забезпечити інші дисципліни обслуговування заявок. Найчастіше застосовують циклічну зміну рівнів пріоритетів, що забезпечує гарантоване обслуговування будь-якого запиту на певному проміжку часі, причому, незалежно від інтенсивності запитів.

Оскільки регістр маски включений в адресний простір процесора, то за умови, що число запитів не перевищує розрядності шини даних (що зазвичай виконується), процесор може за одне звернення до цього регістра записати в нього будь-яку маску.

До недоліків централізованих КПП слід віднести:

– велику кількість ліній запитів в шині управління (що дорівнює кількості ЗП);

– обмеження на максимальне число джерел переривань;

– можлива неоднорідність процесорних модулів.

Останній з вказаних недоліків обумовлений тим, що в централізованих системах приймачем переривань зазвичай є управляючий процесор. Отже, за необхідності використання в системі резерву апаратури на рівні модулів буде потрібно різні типи резерву.

Вказані недоліки обмежують можливості застосування цього способу при побудові однорідної модульної МПС.

Під час використання розподіленого контролера переривань (рис. 5.3) все ЗП повинні бути підключені до магістралі процесора. Це обумовлено тим, що вектор переривання на шину даних в даному випадку видає сам активний ЗП.

До складу кожного ЗП включений блок контролера переривань БКП, який видає сигнал запиту *IRQ*, на загальну лінію *IRQ*. Технологічні особливості елементної бази повинні допускати таке об'єднання виходів елементів (наприклад, використовуються елементи з відкритим колектором, а сигнали запитів мають активний

низький рівень). У відповідь сигнал процесора *IACK* розповсюджується послідовно через елементи БКП, створюючи так званий пріоритетний ланцюжок або "гірлянду" (*daisy chain*). Елементи ланцюжка в кожному БКП пропускають сигнал *IACK* або розривають ланцюжок. Пріоритетний ланцюжок розривається на першому (по шляху розповсюдження сигналу) активному ЗП, який виставив сигнал запиту *IRQ*. Даний активний ЗП видає на шину даних вектор переривання, який приймається процесором.

Приклад побудови БКП показаний на рис. 5.4.

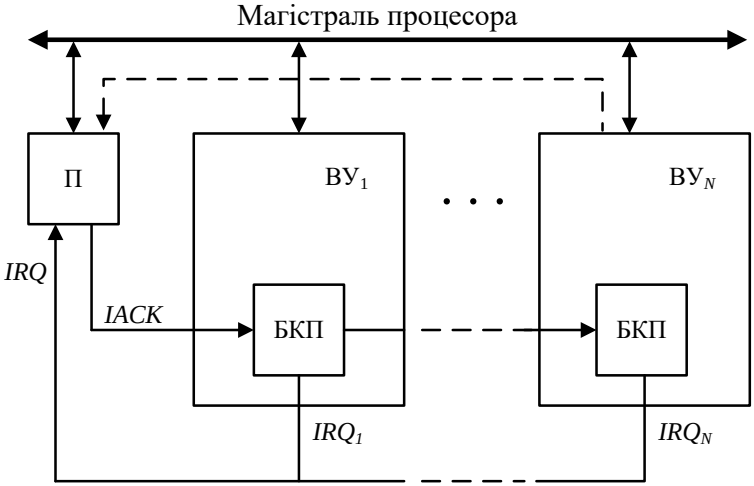


Рис. 5.3. Система с децентралізованим КПП

Блок БКП містить доступні для процесора регістр стану (РС) і регістр вектора (РВ). Під час ініціалізації режиму роботи системи процесор записує в регістр РВ вектор переривання, а в регістр РС – біт дозволу переривання (ДП). Якщо ЗП готовий до взаємодії з процесором, то в регістрі РС встановлюється біт готовності «Г». Це встановлення виконується засобами внутрішнього управління ЗП. За збігу сигналів «Г» і «ДП» формується запит *IRQ*, який через елемент узгодження поступає на лінію *IRQ*. Вхідний для кожного блоку сигнал *IACKin* передається на вихід *IACKout* або забезпечує видачу вектора через буфер вектора (БВ), що визначається значенням *IRQ*.

До достоїнств розподілених КПП можна віднести:

– невелику кількість ліній зв'язку в шині управління;

– простоту нарощування числа ЗП.

Завдяки вказаним достоїнствам розподілені контролери находять широке застосування в МПС системах.

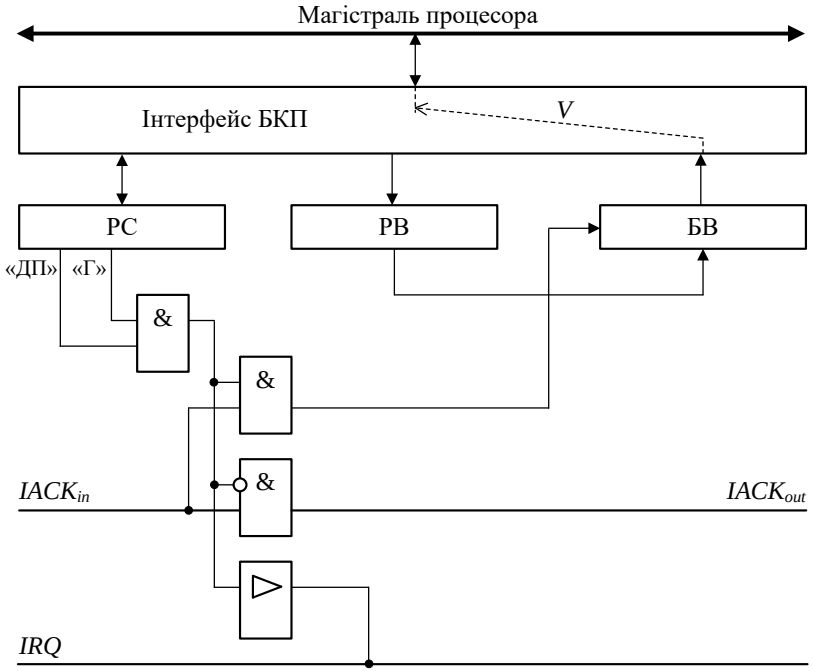


Рис. 5.4. Блок контролера переривань розподіленого КПП

Недоліками розподілених КПП є:

– велика кількість звернень процесора до магістралі під час ініціалізації системи;

– використання фіксованих рівнів пріоритетів запитів, які не можна змінювати динамічно.

Перший недолік обумовлений необхідністю запису біта дозволу переривання в кожен блок окремо. Коли ініціалізація режимів здійснюється рідко, цей недолік можна вважати неістотним.

Використання фіксованих рівнів пріоритетів не забезпечує гарантованого обслуговування заявок на певному відрізку часу. Заявки з низьким рівнем пріоритету за великої інтенсивності заявок з високими пріоритетами можуть не виконуватися тривалий час. Ця

обставина може привести до уповільнення обчислювального процесу, а іноді – до тупикової ситуації.

Обидва розглянуті підходи до реалізації зовнішніх векторних переривань можуть поєднуватися. Наприклад, декілька пристроїв об'єднуються ланцюжком «гірлянди», а декілька ланцюжків, у свою чергу, замикаються на централізованому контролері.

5.2. Умовне графічне позначення мікросхем

Для реалізації централізованого контролеру пріоритетних переривань в МПС можуть бути застосовані мікросхеми КМ1804ВН1 – восьмирозрядна схема пріоритетного переривання, що мікропрограмується та нарощується, та мікросхема КМ1804ВР3.

Мікросхема КМ1804ВР3 є керованим шифратором вісім входів та на три виходи і призначена для спільної роботи з мікросхемами КМ1804ВН1 у складі КПП із різною кількістю входів. Для побудови КПП, що обробляє запити від восьми зовнішніх пристроїв потрібна одна мікросхема КМ1804ВН1. Одна мікросхема КМ1804ВР3 забезпечує прийом і кодування сигналів для восьми мікросхем КМ1804ВН1, таким чином можна побудувати КПП до 64 входів. Для більш складного пристрою потрібне застосування декількох мікросхем КМ1804ВР3.

На рис. 5.5 наведене умовне графічне позначення мікросхеми КМ1804ВН1. Нижче наведені символічні імена виходів мікросхеми.

GND – загальний;
Vcc – напруга живлення +5В;
COMO – управління режимом;
DEINR – заборона переривання;
CRO – перенос із попередньої групи;
EWRSA – дозвіл запису стану;
EINS – дозвіл мікрокоманди;
DES – послідовна заборона;
DEP – паралельна заборона;
RQINR – запит переривання;
CR2 – перенос в наступну групу;
MK [7..0] – маска;
INR [7..0] – запроси переривання;
INS [3..0] – мікрокоманда;
SA [2..0] – поточний стан;

VEC [2..0] – вектор;

CLK – тактовий сигнал.

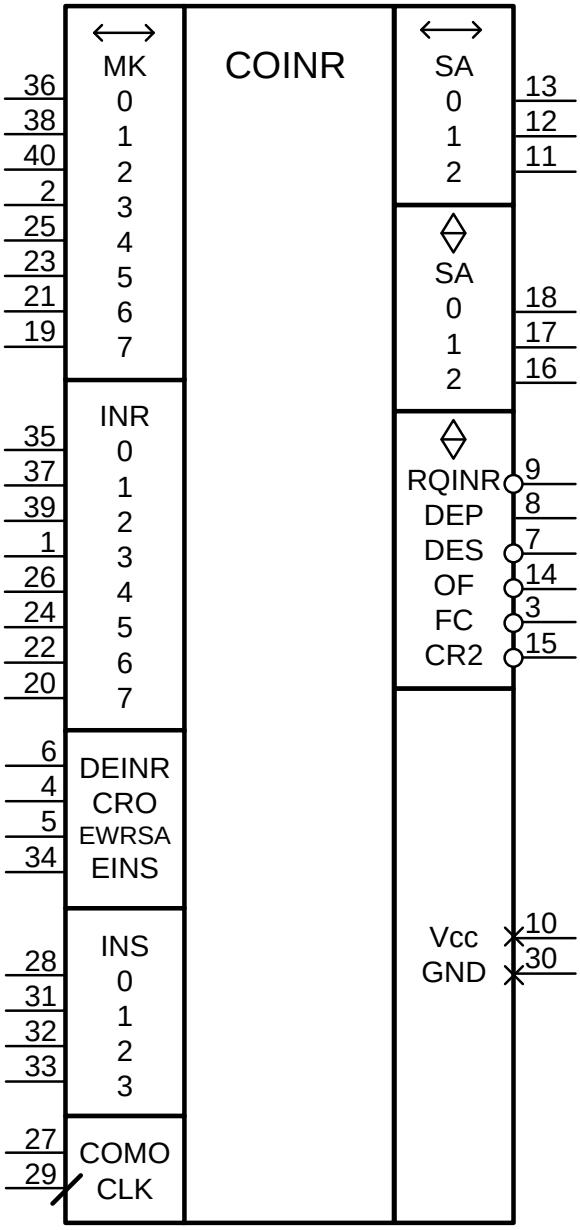


Рис.5.5. Умовне графічне позначення мікросхеми КМ1804ВН1

На рис. 5.6 наведене умовне графічне позначення мікросхеми КМ1804ВР3. Нижче наведені символічні імена виходів мікросхеми.

GND	–	загальний;
Vcc	–	напруга живлення +5В;
EEX1	–	управляючий вхід
EEX2	–	управляючий вихід
D0 – D7	–	вхідна група сигналів
Z0 – Z2	–	вихідна група сигналів

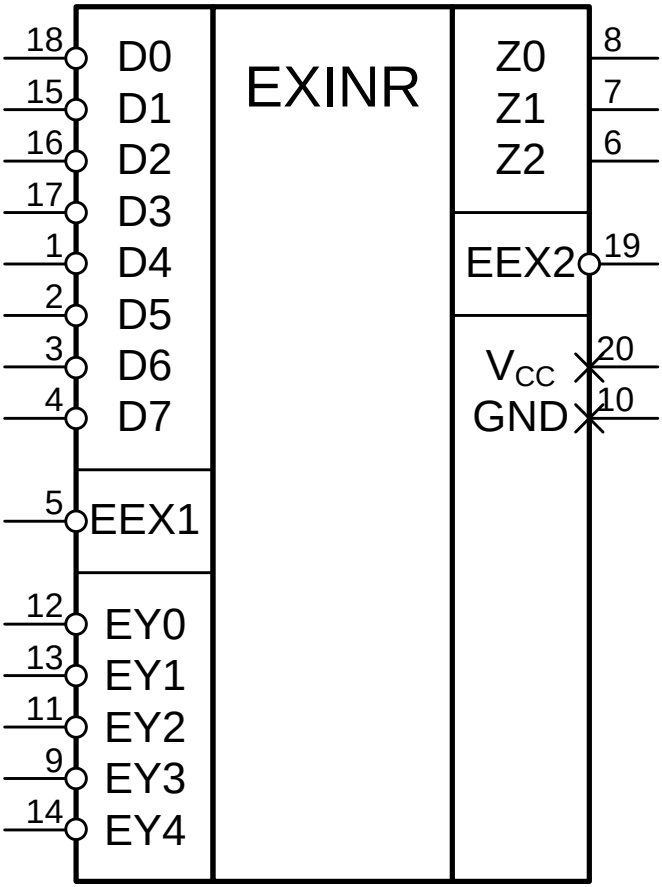


Рис.5.6. Умовне графічне позначення мікросхеми КМ1804ВР3

Таблиця 5.1. Відповідність вхідних та вихідних сигналів

Вхідні сигнали									**Вихідні сигнали**			
*EEX*1	*D*0	*D*1	*D*2	*D*3	*D*4	*D*5	*D*6	*D*7	*Z*2	*Z*1	*Z*0	*EEX*2
1	*	*	*	*	*	*	*	*	0	0	0	1
0	1	1	1	1	1	1	1	1	0	0	0	0
0	*	*	*	*	*	*	*	0	1	1	1	1
0	*	*	*	*	*	*	0	1	1	1	0	1
0	*	*	*	*	*	0	1	1	1	0	1	1
0	*	*	*	*	0	1	1	1	1	0	0	1
0	*	*	*	0	1	1	1	1	0	1	1	1
0	*	*	0	1	1	1	1	1	0	1	0	1
0	*	0	1	1	1	1	1	1	0	0	1	1
0	0	1	1	1	1	1	1	1	0	0	0	1

Примітка: * – будь-який стан входу

Таблиця 5.2. Відповідність вхідних та вихідних сигналів

Вхідні сигнали					**Вихідні сигнали**		
*EZ*1	*EZ*2	*EZ*3	*EZ*4	*EZ*5	*Z*2	Z1	*Z*0
1	1	0	0	0	Дозволено		
0	*	*	*	*	*Z*	*Z*	*Z*
*	0	*	*	*	*Z*	*Z*	*Z*
*	*	1	*	*	*Z*	*Z*	*Z*
*	*	*	1	*	*Z*	*Z*	*Z*
*	*	*	*	1	*Z*	*Z*	*Z*

Примітка: * – будь-який стан входу, *Z* – стан «виключено»

5.3. Реалізація переривань в мікропроцесорних системах

Схеми проектування контролерів пріоритетних переривань приведені на рис. 5.7 –5.12

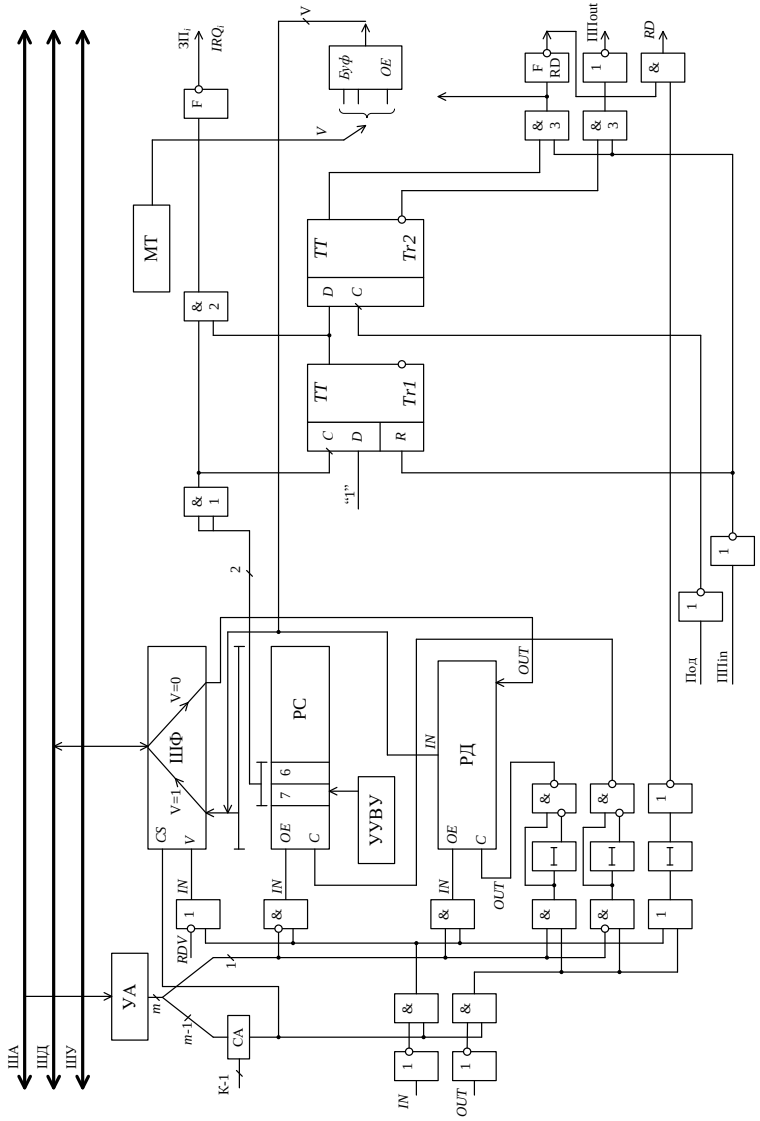


Рис. 5.7. Структурна схема розподіленого КПП

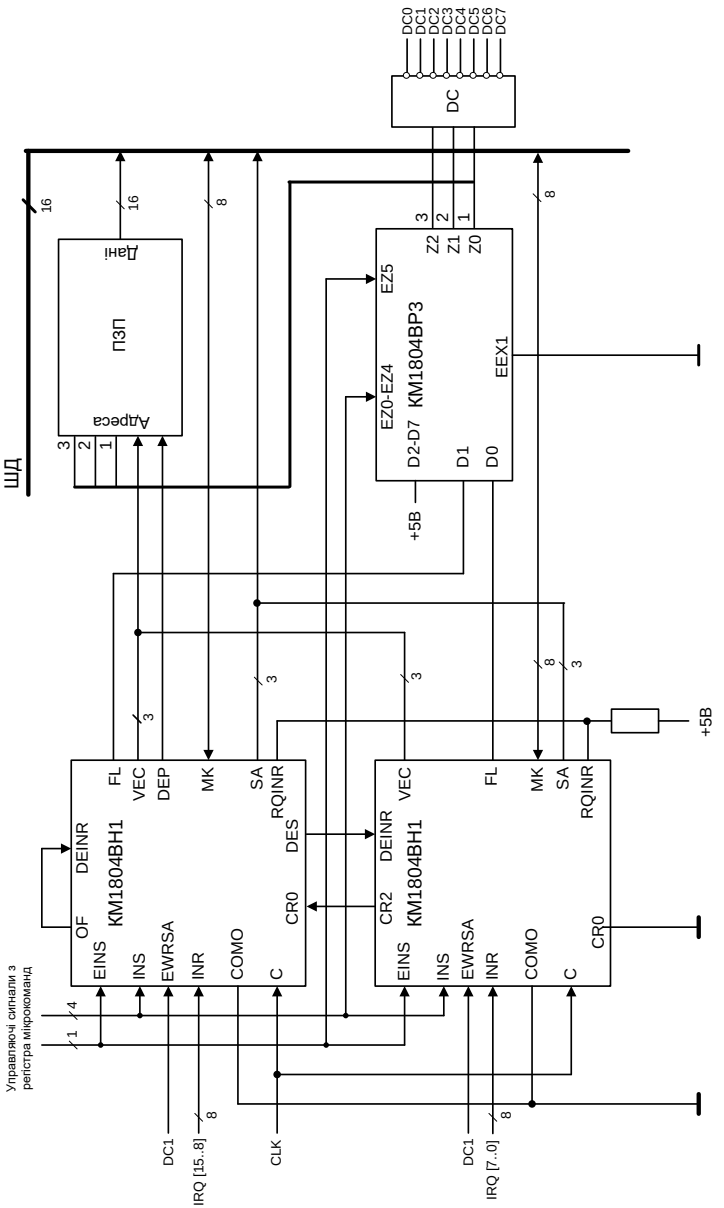


Рис. 5.8. Структурно-функціональна схема централізованого КПП для обслуговування запитів від шістнадцяти зовнішніх пристроїв

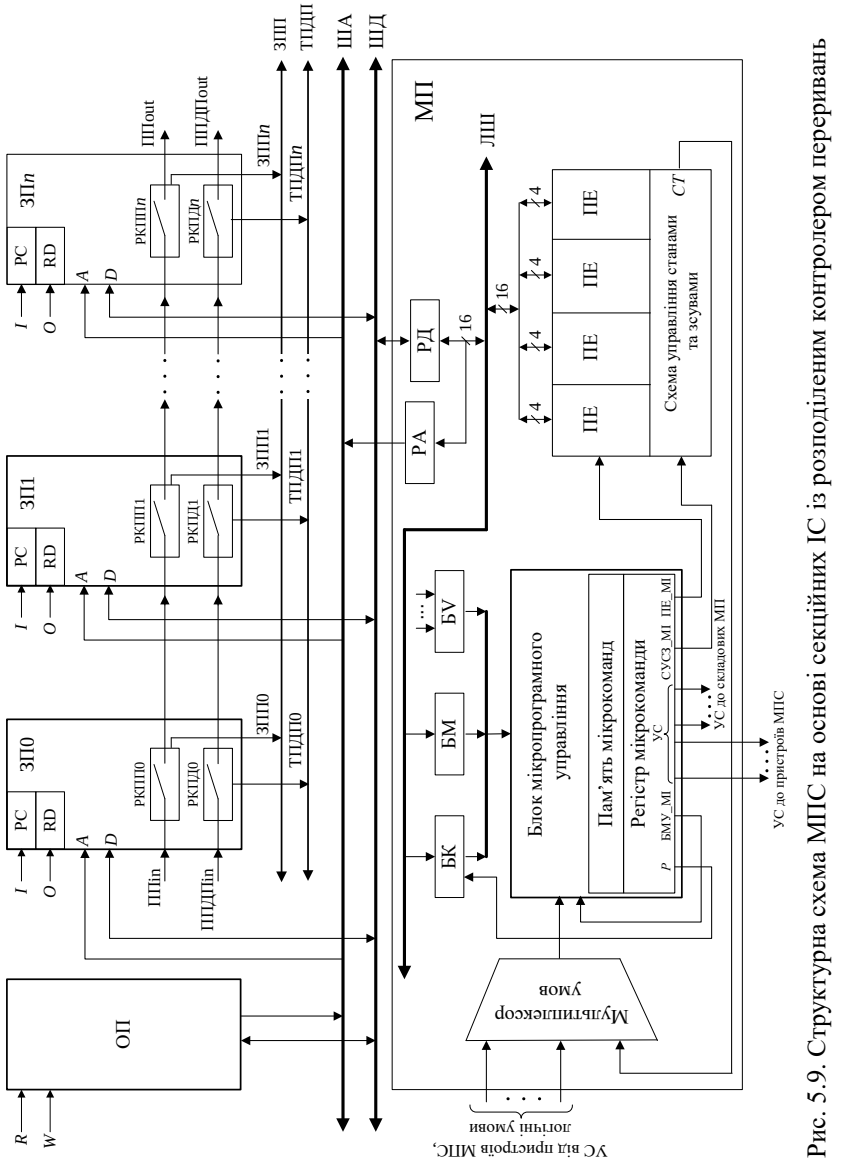


Рис. 5.9. Структурна схема МПС на основі секційних ІС із розподіленим контролером переривань

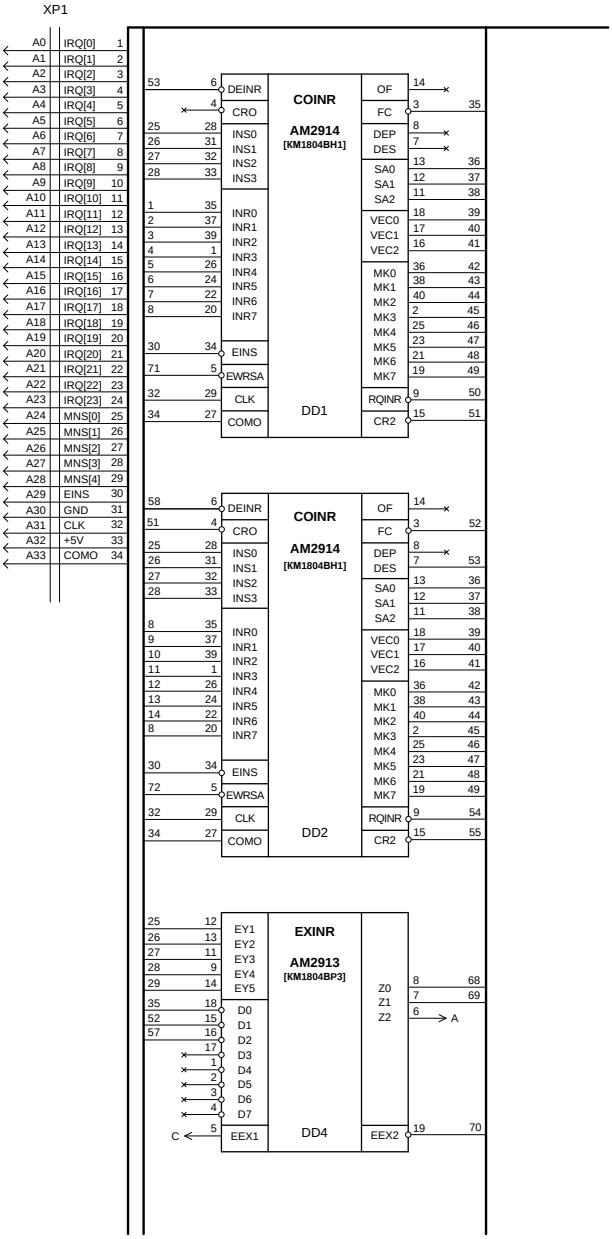
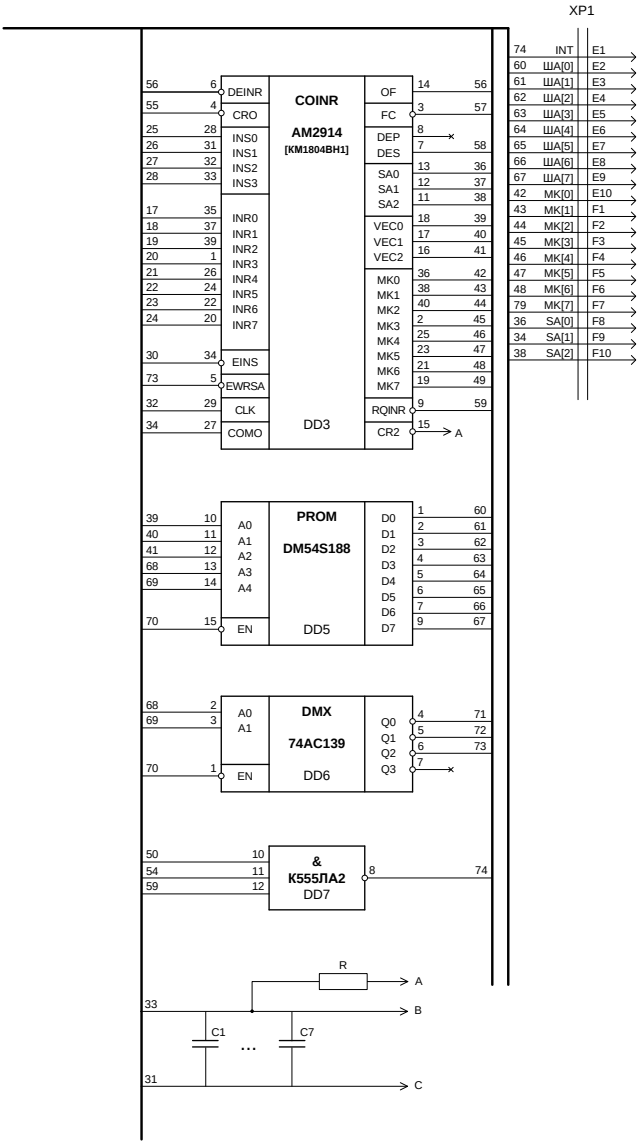


Рис. 5.10. Принципова схема централізованого КПП для обслуговування запитів від двадцяти чотирьох зовнішніх пристроїв



1. Виводи 10 мікросхем DD1 – DD3, 20 мікросхеми DD4,
16 мікросхем DD5 і DD6, 20 мікросхеми DD7 підключити до шини A (+5В)

2. Виводи 30 мікросхем DD1 – DD3, 10 мікросхеми DD4,
8 мікросхем DD5 і DD6, 7 мікросхеми DD7 підключити до шини B (Загальн.)

Продовження рис. 5.10

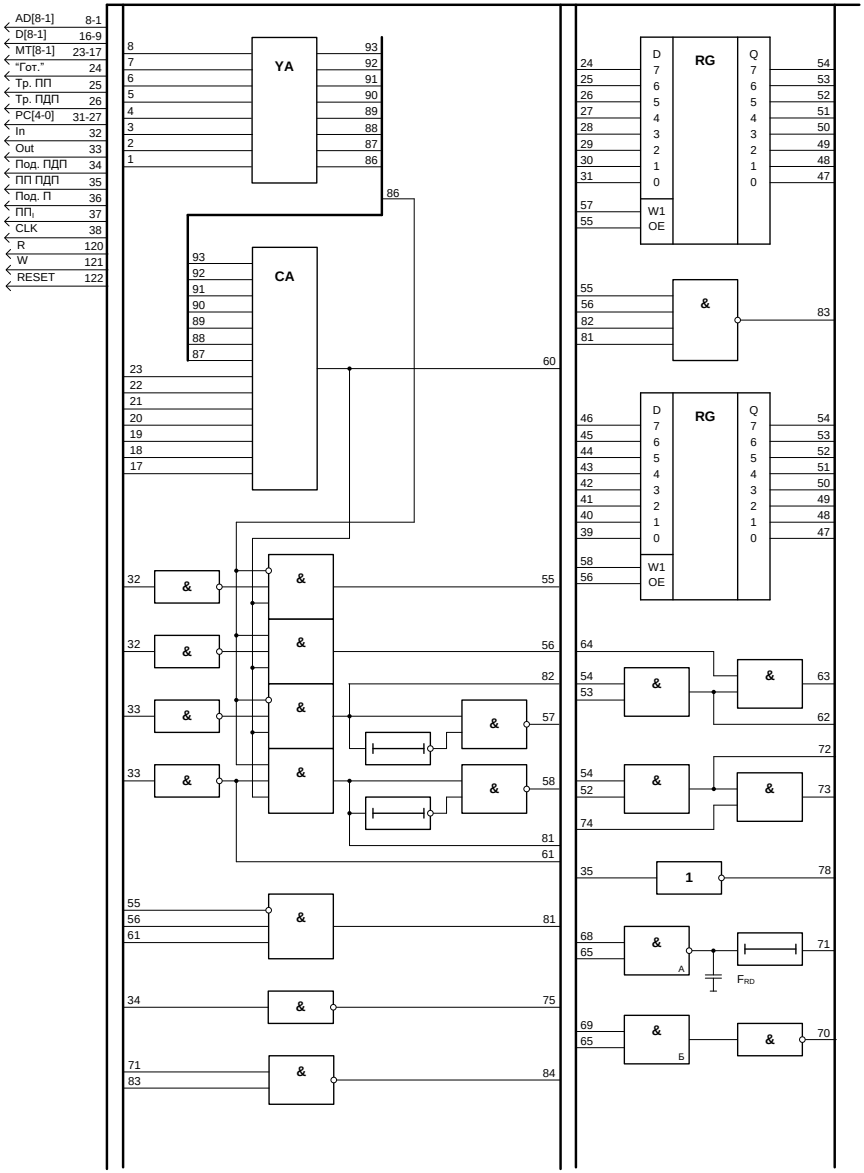
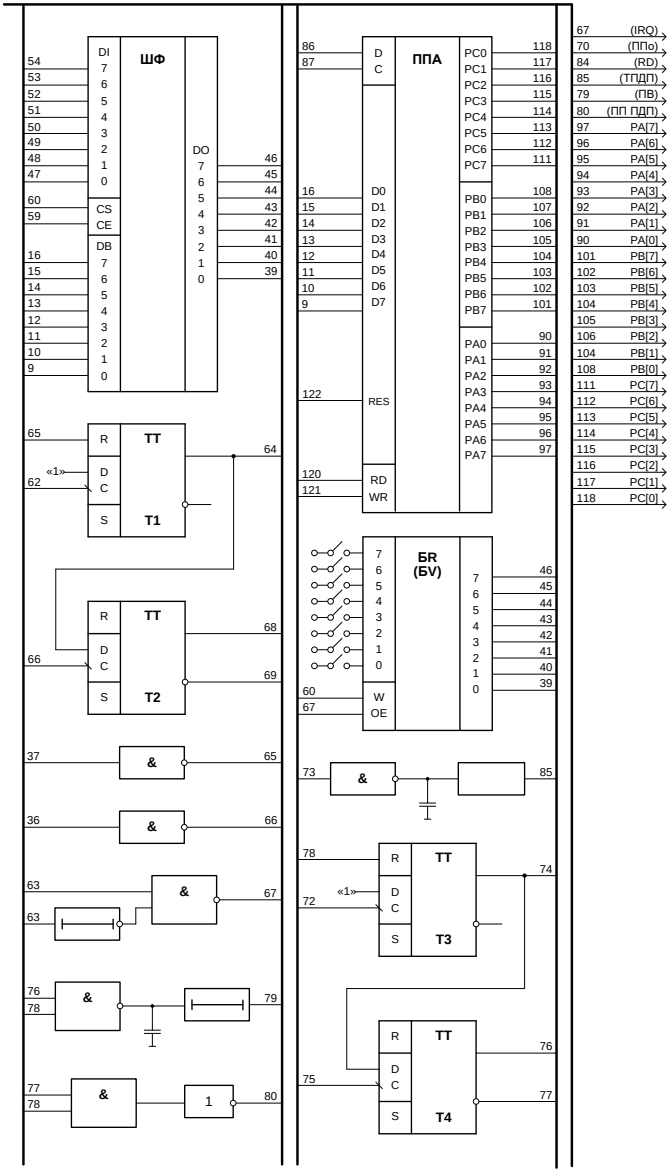


Рис. 5.11. Схема електрична принципова фрагменту МПС, що складається з розподіленого КПП, інтерфейсу ЗП, контролеру ПДП та мікросхеми 580ВВ55 для організації додаткових портів.



Продовження рис. 5.11

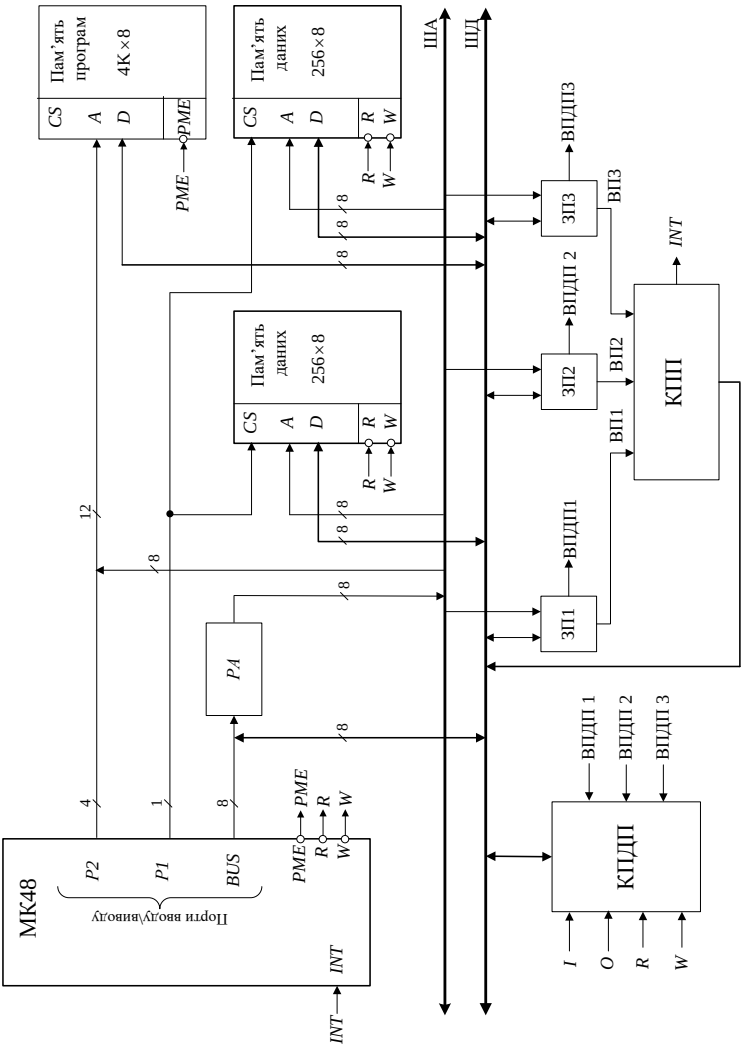


Рис.5.12. Мікропроцесорна система на основі МК із централізованим контролером переривань

ПРАКТИКУМ

6. МОДУЛЬ 1

МІКРОПРОЦЕСОРНІ СИСТЕМИ НА ОСНОВІ СЕКЦІОНОВАНИХ ІНТЕГРАЛЬНИХ СХЕМ

6.1. Лабораторний практикум

6.1.1. Практична робота 1.1

ПОБУДОВА БЛОКІВ ОБРОБКИ ДАНИХ НА СЕКЦІОНОВАНИХ ІНТЕГРАЛЬНИХ СХЕМАХ

Мета роботи: Вивчення схемотехнічних особливостей, системи мікрооперацій ІС К1804ВС1 та К1804ВР2 і принципів побудови блоків обробки даних

Теоретичні відомості: 2.2 – 2.4

Підготовка до виконання практичної роботи

1. Вивчити описи ІС ВС1 та ВР2 і способи побудови операційних блоків

2. Побудувати структурні схеми шістнадцятирозрядних операційних блоків із застосуванням та без застосування ІС ВР2. Для кожного операційного блоку визначити за допомогою часових діаграм мінімальну тривалість такту.

3. Записати номер варіанту (номер залікової книжки) у двійковому поданні і виділити шість молодших розрядів $h_6 - h_1$. За отриманими значеннями двійкових розрядів вибрати варіанти завдань.

4. Розробити мікроалгоритм обчислення функції, що задана у табл. 6.1, забезпечивши формування результату у регістрі вказаному у табл. 6.2. Для подання вихідних даних застосувати доповнювальний код. Під час розробки мікроалгоритму передбачити формування на довільному виході операційного блоку ознаки, що вказує на наявність хоча би в одному розряді результату одиниці.

5. Скласти мікропрограму у кодах мікрокоманд.

6. Виконати числовий приклад застосувавши дані з табл. 6.3. Скласти цифрову діаграму стану регістрів.

7. Вивчити загальні положення до виконання практичних робіт із застосуванням лабораторного комплексу *COMPLEX*. Вказівки до використання мікроасемблеру наведені у додатку А.

Таблиця 6.1. Варіанти завдань

h_2	h_1	**Функція**
0	0	$F = 16(X1 + X2 - 1) \,\&\, (X3 - X4)/2$
0	0	$F = 8(X1 \vee X2) + (X3 - 1 - X4)/16$
1	1	$F = 16[(X1 - 1) \,\&\, X2) - (X3 + X4)/4$
1	1	$F = 8(X1 - X2) + (X3 \oplus X4 - 1)/16$

Таблиця 6.2. Варіанти завдань

h_2	h_4	**Регістр**
0	0	*RQ*
0	0	*R*4
1	1	*R*5
1	1	*R*6

Таблиця 6.3. Варіанти завдань

h_2	h_1	***X*1**	***X*2**	***X*3**	***X*4**
0	0	–17	12	17	3
0	0	12	2	–7	15
1	1	18	–5	23	11
1	1	–9	10	31	–21

Порядок виконання роботи

1. Налагодити розроблену мікропрограму за допомогою лабораторного комплексу *COMPLEX*. Перевірити правильність обчислення заданої функції на числових прикладах.
2. Зробити висновки.

Зміст звіту

Звіт повинен містити короткі теоретичні відомості, структурну схему блоку обробки даних, схему алгоритму, налагоджену мікропрограму у кодах мікропрограм та із застосуванням символічного мікро асемблеру, висновки за роботою.

Контрольні питання

1. Які мікрооперації можна суміщати в розглянутій системі?
2. Приведіть структуру і функціональне призначення БОД.
3. Приведіть структуру мікрокоманди для БОД і функціональне призначення керуючих сигналів для БОД.
4. Охарактеризуйте основні способи додавання та віднімання чисел в ЕОМ. Опишіть загальний склад устаткування, необхідний для реалізації операцій додавання та віднімання в ЕОМ.
5. Що таке арифметичний та логічний зсув? Як забезпечити арифметичний та логічний зсув слів подвоєної довжини?
6. Наведіть структуру і функціональне призначення СУСЗ.
7. Які мікрооперації реалізуються в АЛБ. Як задати в мікропрограмі початкові значення в регістрах АЛБ.
8. Приведіть структуру МК для АЛБ і призначення управляючих сигналів.
9. Наведіть структуру і функціональне призначення СУСЗ.
10. Які мікрооперації виконуються над ознаками результату в СУСЗ? Які типи зсувів забезпечує СУСЗ?
11. Приведіть структуру МК для СУСЗ і призначення управляючих сигналів.

6.1.2. Практична робота 1.2

ВИВЧЕННЯ БЛОКУ МІКРОПРОГРАМНОГО УПРАВЛІННЯ

Мета роботи: Вивчення блоку мікропрограмного управління, а також способи реалізації типових фрагментів мікропрограм та способів розгалуження мікропрограм.

Теоретичні відомості: 2.3, 2.5

Підготовка до виконання практичної роботи

1. Вивчити архітектуру БМУ на основі ІС ВУ4. Побудувати структурно-функціональну схему БМУ.
2. Записати номер варіанту (номер залікової книжки) у двійковому поданні і виділити шість молодших розрядів $h_6 - h_1$. За отриманими значеннями двійкових розрядів вибрати варіант виконання практичної роботи.
3. У відповідності з мікроалгоритмом, приведеним на рис. 6.1, *а* розробити мікропрограму. Мікроалгоритми на рис. 6.1, *б* і рис. 6.1, *в* реалізувати у вигляді мікропідпрограм. Забезпечити звернення із основної мікропідпрограми (рис. 6.1, *а*) до першої мікропідпрограми

(рис. 6.1, *б*) у точках, визначених табл. 6.4, та звернення цієї мікропідпрограми до другої мікропідпрограми (рис. 6.1, *б*) із точки, визначеної табл. 6.5. За необхідності у мікроалгоритми можна вводити додаткові вершини.

4. Розробити мікропрограму у кодах мікрокоманд.

Таблиця 6.4. Варіанти завдань

h_4	h_5	h_1	**Точка переходу на мікропідпрограму**
0	0	0	I
0	0	1	II
0	1	0	III
0	1	1	IV
1	0	0	V
1	0	1	VI
1	1	0	VII
1	1	1	VIII

Таблиця 6.5. Варіанти завдань

h_2	h_1	**Точка переходу на мікропідпрограму**
0	0	IX
0	0	X
1	1	XI
1	1	XII

Порядок виконання роботи

1. Налагодити розроблену мікропрограму за допомогою лабораторного комплексу *COMPLEX*. Перевірити правильність обчислення заданої функції на числових прикладах.
2. Зробити висновки.

Зміст звіту

Звіт повинен містити короткі теоретичні відомості, структурну схему блоку мікро програмного управління, схему алгоритму, налагоджену мікропрограму у кодах мікропрограм та із застосуванням символічного мікро асемблеру, висновки за роботою.

Контрольні питання

1. Поясніть призначення, склад та принцип функціонування ІС ВУ4.
2. Охарактеризуйте виводи мікросхеми ВУ4.
3. Як побудувати БМУ на ІС ВУ4?
4. Які фактори визначають довжину слова мікрокоманди?
5. Приведіть загальну структуру БМУ в обчислювальній системі. Опишіть функціональне призначення кожного елемента.

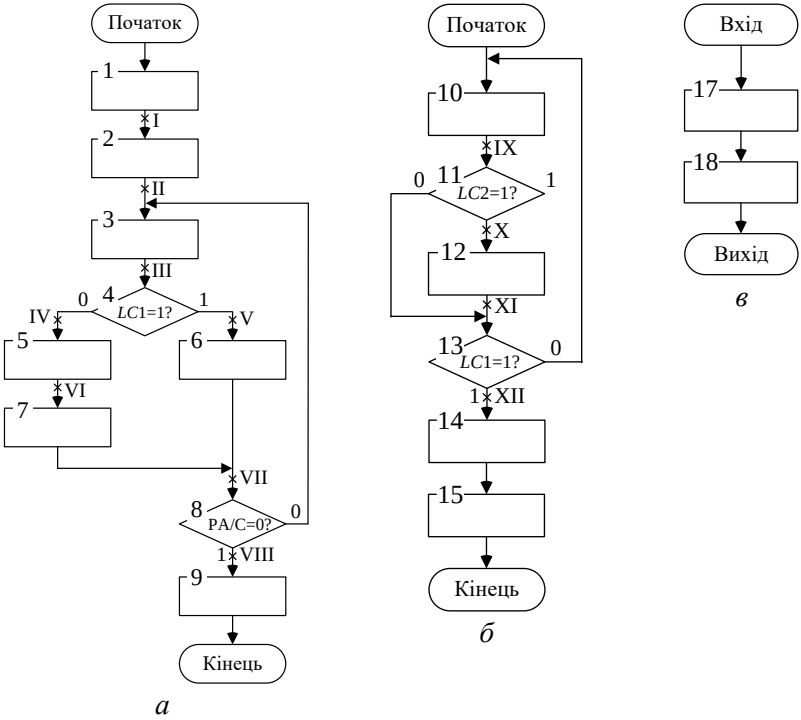


Рис. 6.1. Вихідні мікроалгоритми

6. Навести класифікації БМК. Поясніть, що розуміють під принципом мікропрограмного управління.
7. Поясніть поняття системи команд високого рівня (асемблера), системи мікрокоманд низького рівня та принципу емуляції – реалізації однієї більш складної системи команд за допомогою більш простої системи команд.
8. Опишіть структуру мікрокоманди для БМУ. Опишіть усі можливі способи формування адреси наступної мікрокоманди.

9. Яка мікропрограма називається лінійної. Наведіть спосіб формування адреси у ПМК для лінійних алгоритмів.

10. Як у лінійній мікропрограмі реалізувати безумовний перехід на нову адресу.

11. Яка мікропрограма називається розгалуженою?

12. Як реалізувати умовні та безумовні переходи, цикли, звернення до мікропідпрограми?

13. Як СУСЗ формує сигнал логічної умови, як його перевірити в БОД? Які сигнали можна підключити до входів *L*1 – *L*6 мультиплексора умов?

14. Які мнемоніки логічних умов використовують у мікрокомандах ФАМ? За допомогою якої директиви здійснюють підключення сигналів до входів L1 – L6 мультиплексора?

6.2. Задачі для самостійного розв'язування

Задача 6.1. Розробити мікроалгоритм і мікропрограму на символічному мікроасемблері для реалізації в БОД заданого арифметичного виразу.

Вихідні дані:

- Вираз для обчислення:

а). $F = 4X + 2Y$
б). $F = 2X - 4Y$
в). $F = 4(X + Y/2)$
г). $F = (2X - Y)/8$
д). $F = 2(X + 2Y - Z)$
е). $F = 2(X - 4Y + Z/2)$
ж). $F = (X + Y/2 - 1 + Z) + 1$
з). $F = (Z + 1 + 2X - Y)/8$

- Місце розташування операндів та результату:

	X	*Y*	*Z*	*F*
а).	*R*1	*R*6	*RQ*	*R*6
б).	*R*11	*RQ*	*R*13	*R*15
в).	*RQ*	R1	*R*14	*R*10
г).	*R*4	*R*7	*R*12	*RQ*

Задача 6.2. Розробити для шістнадцятирозрядного БОД мікропрограму обчислення значення функції *F*. Для обчислення значення *D* реалізувати мікропідпрограму.

Вихідні дані:

- Вираз для обчислення:

а). $F = Z + X + D$; $D = 2A + C$;
б). $F = Z + X - D$; $D = 4A + C$;
в). $F = 4Z - X - D$; $D = A/4 + C$;
г). $F = Z + X + 2D$; $D = C - A/2$.

- Місце розташування результату:

д). *RQ;*
е). *R*13;
ж). *R*14;
з). *R*12.

Задача 6.3. Розробити для шістнадцятирозрядного БОД мікропрограму обчислення значення функції *F*. Для обчислення застосувати цикли.

Вихідні дані:

- Вираз для обчислення:

а). $F = 0{,}5C \times 12 + A$;
б). $F = A \times 7 + 4C$;
в). $F = A \times 9 + C$;
г). $F = 0{,}5C \times 10 - 2A$.

- Місце розташування операндів та результату:

д). *RQ*;
е). *R*0;
ж). *R*1;
з). *R*12.

Задача 6.4. Розробити для шістнадцятирозрядного БОД мікропрограму виконання умовного переходу на довільну мітку за заданою ознакою. Рахувати, що слово для аналізу умови знаходиться в акумуляторі *R*15.

Вихідні дані:

- Умова переходу:

а). Вміст акумулятора дорівнює нулю;
б). Третій розряд акумулятора дорівнює нулю;
в). Вміст акумулятора не дорівнює нулю;
г). Число в акумуляторі більш ніж десять;
д). Акумулятор містить парну кількість одиниць;
е). Число в акумуляторі додатне;
ж). Молодший розряд акумулятора дорівнює одиниці;
з). Акумулятор містить непарне число.

Задача 6.5. Розробити для шістнадцятирозрядного БОД мікропрограму виконання заданої операції слова заданої довжини.

Вихідні дані:

- Операція: а). зсув вправо;
  б). зсув вліво;
  в). додавання;
  г). віднімання.
- Довжина слова: д). 32 розряди;
  е). 64 розряди.

Задача 6.6. Розробити для шістнадцятирозрядного БОД мікропрограму виконання операції додавання чисел A і B у заданому машинному двійковому коді. Операцію переводу чисел у заданий двійковий код реалізувати у вигляді мікропідпрограми.

Вихідні дані:

- Двійковий код: а). обернений;
  б). доповнювальний;
- Значення чисел: в). $A = -38$, $B = 13$;
  г). $A = -19$, $B = -23$;
  д). $A = 41$, $B = -117$;
  е). $A = -29$, $B = 143$;
- Довжина слова: ж).16 розрядів;
  з). 32 розряди;
  к). 64 розряди.

Задача 6.7. Розробити для МПС на комплекті 1804 мікроалгоритм і мікропрограму обробки переривань, яка забезпечує перехід на першу команду відповідної підпрограми. Вважати, що в системі використовується розподілений арбітр переривань. Адреси фонової і програми обробки переривань починаються з адрес *A*0*H* і *E*0*H*, відповідно.

7. МОДУЛЬ 2

ОДНОКРИСТАЛЬНИЙ МІКРОКОНТРОЛЕР КР1816ВЕ48

7.1. Лабораторний практикум

7.1.1. Практична робота 2.1

ВИВЧЕННЯ СИСТЕМИ КОМАНД МІКРОКОНТРОЛЕРА КР1816ВЕ48

***Мета роботи*:** Вивчення структурної схеми МК48, системи та форматів команд. Отримання навиків складання алгоритмів та розробки програм виконання арифметичних операцій для мікроконтролера МК48.

Теоретичні відомості: 3.2, 3.3, 3.4

Додаткові теоретичні відомості

Структура програми на мові асемблера МК48

Текст програми на мові асемблера МК48 має певну структуру. Кожна команда (і псевдокоманда) записуються в рядок наступної конструкції:

```
МІТКА: ОПЕРАЦІЯ ОПЕРАНД(И)    ; КОМЕНТАР
```

Поля можуть відділятися один від одного довільним числом пробілів.

Мітка

У полі мітки розміщується символічне ім'я комірки пам'яті, в якій зберігається відмічена команда або операнд. Мітка є буквено-цифровою комбінацією, що починається з букви. Використовуються тільки букви латинського алфавіту. Асемблер МК48 допускає використання в мітках символу підкреслення (_). Довжина мітки не повинна перевищувати шість символів. Мітка завжди завершується двокрапкою (:).

Операція

В полі операції записується мнемонічне позначення команди МК48 або псевдокоманди асемблера, яке являється скороченням (абревіатурою) повної англійської назви виконуючої дії.

Наприклад:

MOV	–	*Move*	–	перемістити;
JMP	–	*Jump*	–	перейти;
DB	–	*Define Byte*	–	визначити байт.

Для МК48 використовується строго визначений та обмежений набір мнемонічних кодів. Будь-який інший набір символів, що розміщений в полі операції, сприймається асемблером як помилковий.

Операнди

В цьому полі визначаються операнд (або операнди), які приймають участь в операції. Команди асемблера можуть не мати операндів або мати один чи два операнди. Операнди розділяються комою (,).

Операнд може бути заданий безпосередньо або його адресою (прямою або непрямою).

Використані в якості операндів символічні імена та мітки повинні бути визначені, а числові значення – подані з вказівкою на систему числення. Для вказівки системи числення використовується суфікс – буква, яка розміщується після числа. Число без суфіксу вважається десятковим. Застосовуються наступні суфікси:

B – двійкова система обчислення;
Q – вісімкова система обчислення;
D – десяткова система обчислення;
H – шістнадцятирічна система обчислення.

Обробка виразів в процесі трансляції

Вираз представляє собою сукупність символічних імен та числових значень, що пов'язані операторами асемблера. Оператори асемблера забезпечують виконання в форматі двобайтних слів наступних операцій:

арифметичних операцій:			логічних операцій:		
+	–	додавання;	ORL	–	АБО;
–	–	віднімання;	ANL	–	І;
*	–	множення;	XRL	–	виключне АБО;
/	–	ділення;	NOT	–	заперечення.
MOD	–	ділення за модулем;			

Наприклад, запис `ADD A,#((NOT 13)+1)` еквівалентний запису `ADD A,#0F3H` та забезпечує додавання до вмісту акумулятора числа (–13), що подане доповнювальним машинним кодом.

Коментар

Поле коментарю застосовується для текстового чи символьного пояснення логічної організації прикладної програми. Поле коментарю ігнорується асемблером, тому тут допускається використання будь-яких символів. За правилами мови асемблера поле коментарію починається після символу крапки з комою (;).

Псевдокоманди асемблера

Спеціальна програма асемблер транслює вихідну програму в об'єктні коди. Асемблер бере на себе багато рутинних задач програміста, таких як присвоєння дійсних адрес, перетворення чисел, присвоєння дійсних значень символьним змінним і таке інше. Програміст вказує асемблеру наступні параметри: початкову адресу прикладної програми, кінець програми асемблера, формати даних. Ця інформація додається у вихідний текст прикладної програми у вигляді псевдокоманд (директив) асемблера, які управляють процесом трансляції і не перетворюються в коди об'єктної програми.

Застосування псевдокоманд:

– псевдокоманда `ORG 10H` задає асемблеру адресу комірки пам'яті 10*h*, в якій розміщується наступна команда прикладної програми;

– псевдокоманда `EQU` будь-якому символічному імені, яке використовується в програмі, ставить у відповідність певний операнд; наприклад, запис `PET EQU 13`, означає, що під час асемблювання символічне ім'я *PET*, буде замінено числом 13.

– псевдокоманда `END` дає асемблеру вказівку закінчити трансляцію.

Мікроконтролер оперує командами чотирьох типів, формат яких наведений на рис. 3.33 (розділ 3.4).

В розділі 3.4 наводяться приклади виконання арифметичних та логічних команд, команд роботи з ознаками, команд зсувів та інших команд, які треба використати при розробці програм в даній практичній роботі.

Підготовка до виконання практичної роботи

1. Вивчити архітектуру і систему команд МК48.

2. Записати номер варіанту (номер залікової книжки) у двійковому поданні і виділити шість молодших розрядів $h_6 - h_1$. Значення операндів обрати з табл. 7.1, які подані в шістнадцятирічній системі числення. Операнди (*X*1 – *X*6) записати в регістри із портів заданих в табл. 7.1.

3. Розробити операційні схеми, цифрові діаграми стану регістрів під час виконання основних операцій для обчислення виразу (табл. 7.2).

4. Розробити для МК48 функціонально-структурний алгоритм обчислення виразу.

5. Розробити програму обчислення виразу на асемблері МК48. Ознаку переносу, яка виникає під час виконання операцій додавання та зсувів, запам'ятовувати в окремий регістр, для отримання правильного результату, що має розмір два байта.

Таблиця 7.1. Варіанти завдань

h_2	h_1	*X*1	*X*2	*X*3	*X*4	*X*5	X6	h_4	h_3	Порт
0	0	33	*4A*	1*C*	06	20	05	0	0	*P*1
0	1	05	2*B*	32	1*A*	04	6*E*	0	1	*P*2
1	0	45	05	3*C*	*A*0	22	12	1	0	*BUS*
1	1	2*C*	03	71	*CA*	11	*FF*	1	1	*P*4,*P*5

Таблиця 7.2. Варіанти завдань

h_5	h_1	Вираз для обчислення
0	0	$F = 4(X1 + X2 - 1) - (X3 + X4) - (X5 \,\&\, X6)/4$
0	1	$F = 4((X1 + X2) \,\&\, (X3 - X4 - 1))/2 - 8(X5 \vee X6)$
1	0	$F = 4(X1 - X2) + (X3 \,\&\, X4) - (X5 - X6 - 1)/16$
1	1	$F = 4(X1 \,\&\, X2) - (X3 - X4 + 1) - (X5 \vee X6 - 1)/2$

Порядок виконання роботи

1. Використовуючи моделюючий комплекс *SCM* МК48 (додаток Б) налагодити розроблену програму:

– сформувати початковий текст програми у вікні екранного редактору комплексу *SCM* МК48, виконати пошук синтаксичних помилок та скомпілювати програму;

– налагодити розроблену програму пристрою управління за допомогою програмно-логічної моделі *SCM* МК48 з отриманням повної інформації про хід виконання програми.

2. Зробити висновки.

Зміст звіту

Звіт повинен містити короткі теоретичні відомості, операційні схеми, цифрові діаграми стану регістрів, схему алгоритму, налагоджену програму на символічному асемблері, висновки за роботою.

Контрольні питання

1. Які способи адресації використовуються під час звернення до регістрів банків регістрів?
2. Яким чином виконують переключення між банками регістрів?
3. У якій області пам'яті розташовуються банки регістрів?
4. Які регістри застосовуються як покажчики адреси під час непрямої адресації?
5. Чому регістр *R5* не може бути застосований як покажчик адреси під час непрямої адресації?
6. Зобразити за допомогою мнемонічної схеми формування адреси для ОЗП.
7. У чому полягає основна відмінність ознак, що входять до складу регістру *PSW*, та інших ознак, що формуються в МК48?
8. Як реалізувати додавання двох двобайтних чисел?

7.1.2. Практична робота 2.2

ВИВЧЕННЯ КОМАНД ПЕРЕДАЧІ УПРАВЛІННЯ МК48

Мета роботи: Вивчення системи команд МК48 і отримання навиків розробки програм, що вміщують команди передачі управління.

Теоретичні відомості: 3.3, 3.4, 3.5.3.

Підготовка до виконання практичної роботи

1. Вивчити архітектуру і систему команд МК48.
2. Записати номер варіанту (номер залікової книжки) у двійковому поданні і виділити шість молодших розрядів $h_6 - h_1$. Значення операндів обрати з табл. 7.3, які подані в

шістнадцятирічній системі числення. Операнди (*X*1 – *X*6) записати в регістри із портів заданих в табл. 7.4. В регістр *R*7 записується кількість ітерацій циклу, яка також задається в таблиці 7.3.

3. Вибрати алгоритм для розробки програми на асемблері за табл. 7.5. Варіанти алгоритмів надані на рис. 7.1.

4. Розробити операційні схеми, цифрові діаграми стану регістрів під час виконання основних операцій для обчислення виразу.

5. Розробити для МК48 функціонально-структурний алгоритм обчислення виразу.

6. Розробити програму обчислення виразу на асемблері МК48. Ознаку переносу, яка виникає під час виконання операцій додавання та зсувів, запам'ятовувати в окремий регістр, для отримання правильного результату, що має розмір два байта.

Таблиця 7.3. Варіанти завдань

h_4	h_1	***X*0**	***X*1**	***X*2**	***X*3**	***X*4**	***X*5**	***X*6**	**Кількість ітерацій циклу**
		***R*0**	***R*1**	***R*2**	***R*3**	***R*4**	***R*5**	***R*6**	***R*7**
0	0	4*A*	*D*3	*E*5	24	06	20	05	3
0	1	2*B*	05	27	22	1*A*	04	6*E*	4
1	0	55	45	29	20	*A*0	02	12	5
1	1	03	5*C*	30	67	0A	11	*A*0	7

Таблиця 7.4. Варіанти завдань

h_5	h_2	**Порти**
0	0	*P*1
0	1	*P*2
1	0	*BUS*
1	1	*P*4,*P*5

Таблиця 7.5. Варіанти завдань

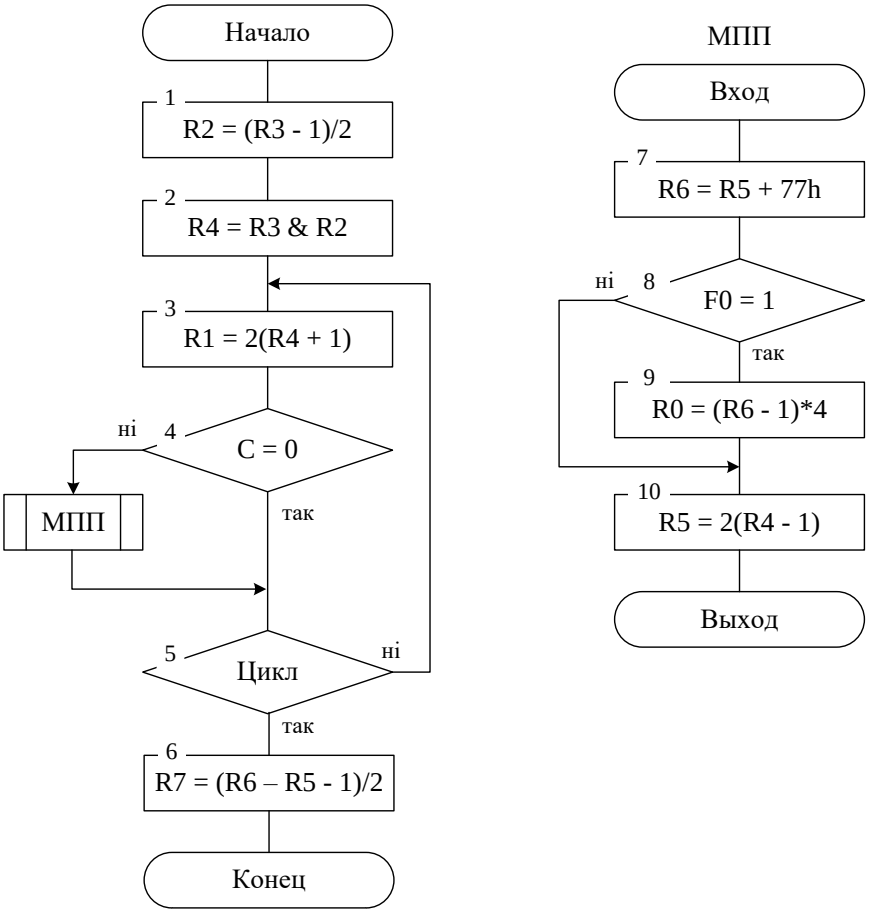
h_2	h_3	**Алгоритм**
0	0	рис. 7.1, *а*
0	1	рис. 7.1, *б*
1	0	рис. 7.1, *в*
1	1	рис. 7.1, *г*

Зміст звіту

Звіт повинен містити короткі теоретичні відомості, операційні схеми, цифрові діаграми стану регістрів, схему алгоритму, налагоджену програму на символічному асемблері, висновки за роботою.

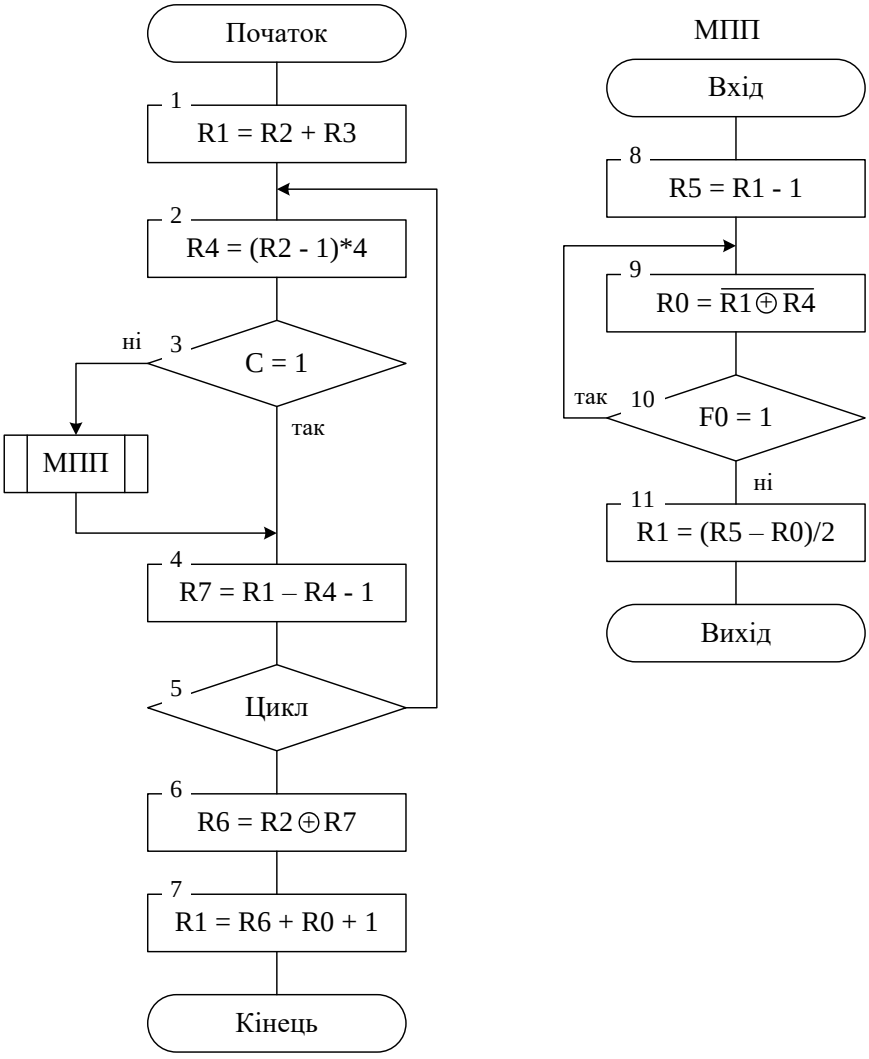
Контрольні питання

1. Які способи адресації використовуються під час звернення до ОЗП?



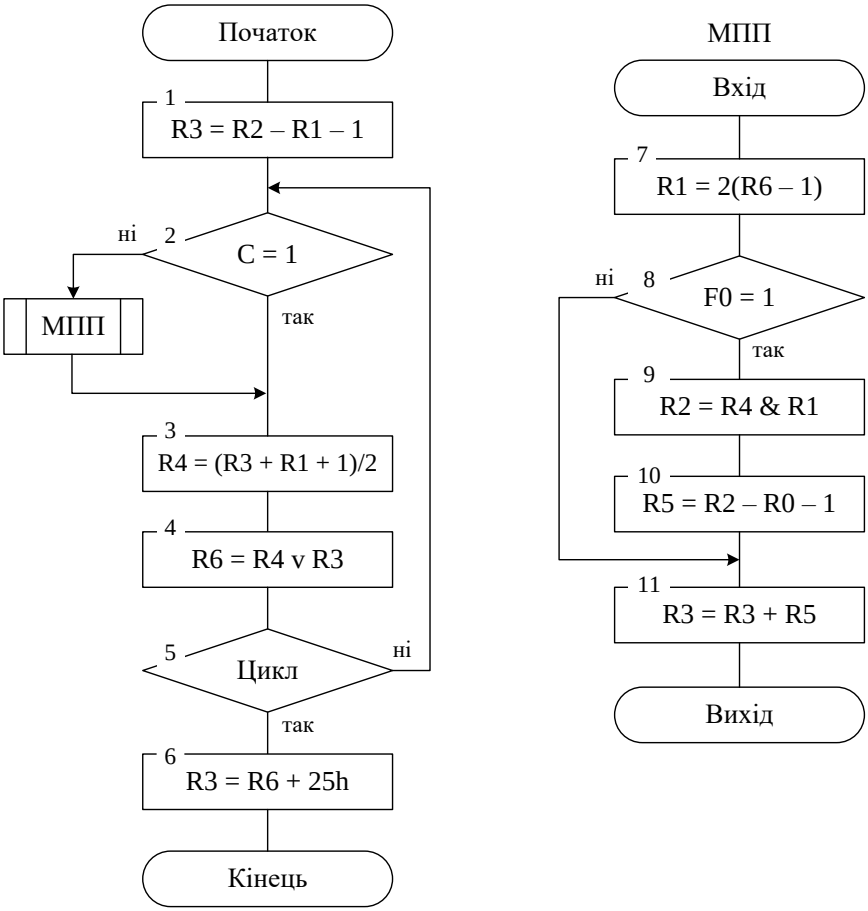
а

Рис. 7.1. Вихідний алгоритм



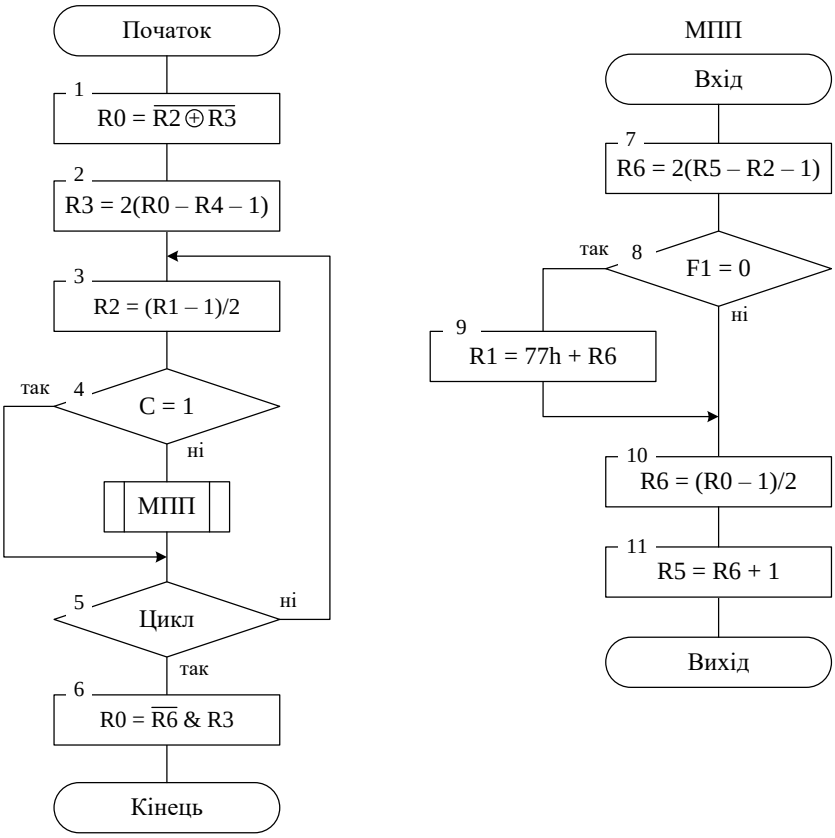
б

Продовження рис. 7.1



в

Продовження рис. 7.1



г

Продовження рис. 7.1

2. Які регістри являються покажчиками адреси за непрямої адресації під час звернення до резидентної та зовнішньої пам'яті даних?

3. Привести приклад зчитування констант з пам'яті програм. Яким чином можна переключатися між банками пам'яті програм?

4. Охарактеризувати команди зчитування з пам'яті програм. Яким чином виконуються умовні та безумовні переходи на підпрограми? Який регістр використовується як покажчик стеку?

5. Яким чином виконується запис інформації в стек? Яка глибина стеку?

6. Яким чином підключаються сторінки зовнішньої пам'яті даних?

7. Привести приклад запису інформації в зовнішню пам'ять даних. Який спосіб адресації при цьому використовується? Які регістри є покажчиками адреси?

8. Привести приклад підключення до МК48 десяти сторінок ЗПД. Показати, як виконується вибір сторінок пам'яті:

а). п'ятої сторінки; в). п'ятнадцятої сторінки;

б). десятої сторінки; г). тридцять другої сторінки.

7.1.3. Практична робота 2.3

ПРОГРАМНЕ ФОРМУВАННЯ ЧАСОВОЇ ЗАТРИМКИ В МК48

Мета роботи: Вивчення структури, режимів роботи, системи команд і отримання навиків розробки програм, що управляють, для мікроконтроллера КМ1816ВЕ48.

Теоретичні відомості: 3.3, 3.4, 3.5.5

Додаткові теоретичні відомості

Формування часової затримка малої тривалості

Процедура реалізації часової затримки основана на методі програмних циклів. При цьому в довільний робочий регістр завантажується число, яке в кожній ітерації циклу зменшується на 1. Так продовжується до тих пір, поки вміст робочого регістра не стане рівним нулю, що інтерпретується програмою як вихід з циклу. Час затримки визначається величиною, завантаженою в робочий регістр, і часом виконання команд, що утворюють програмний цикл.

Для отримання заданої часової затримки необхідно визначити значення *X*, яке завантажується в робочий регістр. Обчислення значення *X* базується на часі виконання команд, що утворюють дану підпрограму. При цьому необхідно враховувати, що команди `MOV` і `RET` виконуються одноразово, а число повторень команди `DJNZ` дорівнює числу *X*. Крім того, звернення до підпрограми часової

затримки здійснюється за командою CALL, час виконання якої також необхідно враховувати під час обчислення часової затримки. При тактовій частоті 6 Мгц кожен машинний цикл виконується за 2,5 мкс. В підпрограму можуть бути включені додаткові команди NOP, час виконання кожною з яких рівне 2,5 мкс.

Формування часової затримка великої тривалості

Недоліком програмного способу реалізації часової затримки є нераціональне використання ресурсів МК48. Апаратурні засоби МК48 дозволяють реалізувати часові затримки на фоні основної програми роботи.

Під час використання таймера можна отримати часові затримки тривалістю від 80 мкс до 20 мс. Наприклад, для реалізації часової затримки 240 мкс необхідно виконати наступні дії:

```
        MOV     A,#FDh  ; завантаження в A (–3)ДК
        MOV     T,A     ; завантаження таймера
        ORL     BUS,…   ; установка управляючого сигналу
        STRT    T       ; запуск таймера
LL1:    JTF     LL2     ; перевірка ознаки переповнення від T
        JMP     LL1;    ; зняття управляючого сигналу
LL2:    ANL     BUS,…   ; зняття управляючого сигналу
```

Поява сигналу переривання від таймера відповідає закінченню часового інтервалу 240 мкс. Погрішність може складати 4-5 мкс.

Підготовка до виконання практичної роботи

1. Вивчити архітектуру і систему команд МК48.

2. Записати номер варіанту (номер залікової книжки) у двійковому поданні і виділити шість молодших розрядів $h_6 - h_1$. За отриманими значеннями двійкових розрядів вибрати варіант алгоритму управління з табл. 7.6. Вихідні алгоритми відповідно до варіантів надані на рис. 7.2. В операторні вершини заданого алгоритму управління записати управляючи сигнали, надані в табл. 7.7.

3. Розробити програму управління для МК48, що реалізує заданий алгоритм. Для вводу/виводу даних використати порти, задані в табл. 7.8. Часові параметри управляючих сигналів задані в табл. 7.9.

4. Побудувати часові діаграми управляючих сигналів для всіх комбінацій значень логічних умов.

Таблиця 7.6. Варіанти завдань

h_2	h_1	**Вихідний алгоритм**
0	0	рис. 7.2, *а*
0	1	рис. 7.2, *б*
1	0	рис. 7.2, *в*
1	1	рис. 7.2, *г*

Таблиця 7.7. Варіанти завдань

h_2	h_3	**Управляючі сигнали**			
		Номер операторної вершини			
		1	2	3	4
0	0	y_1y_2	y_1y_3	y_4y_5	y_3
0	1	y_5	$y_1y_3y_4$	y_1	y_3y_4
1	0	$y_1y_2y_3$	y_2	y_4y_1	y_4
1	1	y_1y_5	y_2	$y_2y_3y_4$	$y_1y_2y_5$

Таблиця 7.8. Варіанти завдань

h_2	h_4	**Порт**
0	0	*BUS*
0	1	*P*1
1	0	*P*2
1	1	*P*4, *P*5

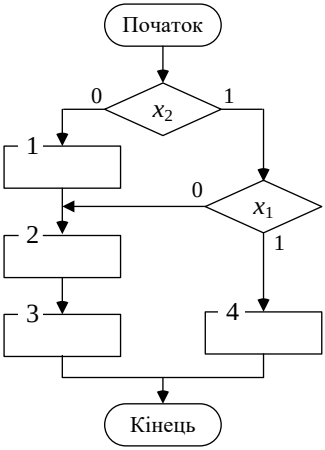
Таблиця 7.9. Варіанти завдань

h_4	h_5	h_1	**Часові затримки**				
			Управляючи сигнали				
			y_1	y_2	y_3	y_4	y_5
0	0	0	720	25	400	10	800
0	0	1	560	12	700	330	280
0	1	0	100	725	15	22	45
0	1	1	40	240	100	50	150
1	0	0	18	280	80	720	60
1	0	1	400	35	800	180	12
1	1	0	560	45	20	150	22
1	1	1	222	15	720	500	560

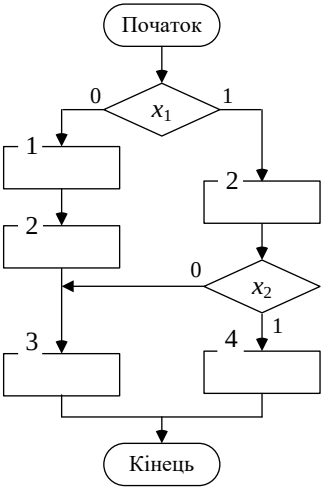
Порядок виконання роботи

1. Використовуючи моделюючий комплекс *SCM* MK48 налагодити розроблену програму:

– сформувати початковий текст програми у вікні екранного редактору комплексу *SCM* MK48, виконати пошук синтаксичних помилок та скомпілювати програму;



а



б

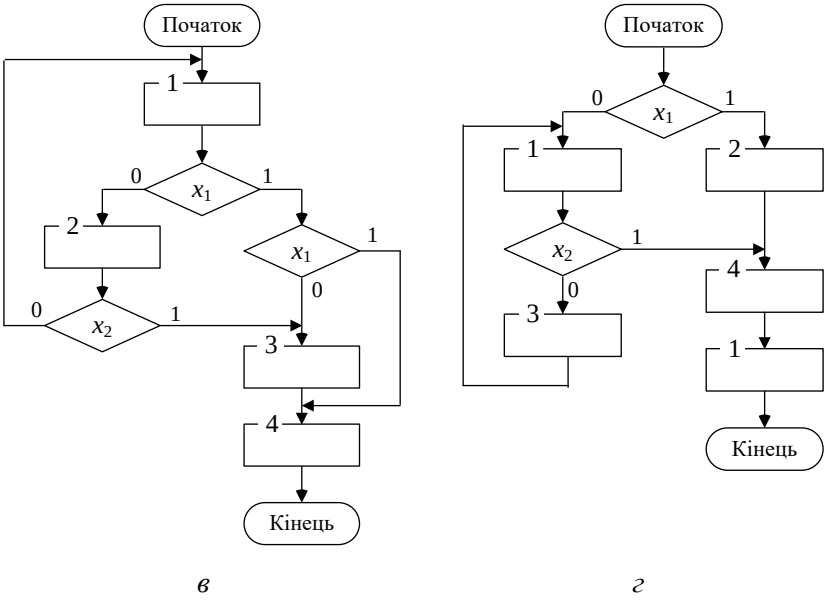


Рис. 7.2. Вихідні мікроалгоритми

– налагодити розроблену програму пристрою управління за допомогою програмно-логічної моделі *SCM* МК48 з отриманням повної інформації про хід виконання програми.

2. Отримати часові діаграми управляючих сигналів на виході заданого порту.

3. Зробити висновки.

Зміст звіту

Звіт повинен містити короткі теоретичні відомості, схему алгоритму, налагоджену програму на символічному асемблері, часові діаграми управляючих сигналів, висновки за роботою.

Контрольні питання

1. Охарактеризуйте основні групи системи команд МК48.
2. Наведіть формат команди МК48.
3. Які способи адресації даних застосовуються в МК48?
4. Наведіть модель програміста МК48.
5. Як здійснюється обмін даними з зовнішніми пристроями?

6. Як налаштувати виходи портів мікроконтролера на введення даних? Приведіть часову діаграму.

7. Розробіть програму формування управляючого сигналу заданої тривалості.

а). 246 мкс; в). 285 мск;

б). 480 мкс; г). 900 мкс.

8. Як здійснити запуск таймера/лічильника в різних режимах?

9. Охарактеризуйте можливості додаткових портів МК48.

10. Приведіть приклади використання в програмі ознак користувача *F*0 і *F*1.

11. Накресліть структурну схему підключення до МК48 зовнішньої пам'яті програм і даних.

12. У чому полягає основна відмінність ознак, що входять до складу регістру *PSW* і ознак, що не входять до складу цього регістру?

13. У якій області пам'яті розміщуються підпрограми обслуговування переривань?

14. Вкажіть призначення кожного виходу мікросхеми мікроконтролера МК48.

7.1.4. Практична робота 2.4

ВИКОНАННЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ В МІКРОКОНТРОЛЕРІ МК48

Мета роботи: Вивчення структури, режимів роботи, системи команд і отримання навиків розробки програм виконання операції множення в мікроконтроллері МК48.

Теоретичні відомості: *3.3, 3.4, 3.5.2, 3.5.4, [1], [15]*.

Підготовка до виконання практичної роботи

1. Записати номер варіанту (номер залікової книжки) у двійковому поданні і виділити шість молодших розрядів $h_6 - h_1$.

2. За отриманими значеннями двійкових розрядів вибрати спосіб множення та розрядність операндів з табл. 7.10. Машинні коди, в яких повинні бути подані операнди *X* та *Y* у вихідному стані, та результат Z наприкінці виконання обчислень, визначені в табл. 7.11, де ПК – прямий код, ДК – доповнювальний код.

3. Розробити операційну схему, цифрову діаграму стану регістрів, схему алгоритму виконання операції множення двійкових чисел $Z = X \times Y$ (де $X < 1, Y < 1$) у заданих машинних кодах. Значення операндів вибрати самостійно.

4. Розробити програму реалізації операції множення на асемблері МК48. Порти для вводу/виводу даних, надані в табл. 7.12.

5. Визначити час виконання операції при опорній частоті F=6МГц.

Таблиця 7.10. Варіанти завдання

h_6	h_5	h_4	**Спосіб множення**	**Розрядність операндів**
0	0	0	1	24
0	0	1	2	16
0	1	0	3	24
0	1	1	4	8
1	0	0	1	24
1	0	1	2	8
1	1	0	3	24
1	1	1	4	16

Таблиця 7.11. Варіанти завдання

h_5	h_3	h_1	***X***	***Y***	***Z***
0	0	0	ДК	ДК	ПК
0	0	1	ПК	ПК	ДК
0	1	0	ПК	ДК	ПК
0	1	1	ДК	ПК	ПК
1	0	0	ПК	ПК	ДК
1	0	1	ДК	ДК	ПК
1	1	0	ДК	ПК	ДК
1	1	1	ПК	ДК	ДК

Таблиця 7.12. Варіанти завдань

h_2	h_4	**Порт**
0	0	*P*4, *P*5
0	1	*BUS*
1	0	*P*1

1	1	*P*2

Порядок виконання роботи

1. Використовуючи моделюючий комплекс *SCM* МК48 налагодити розроблену програму:

– сформувати початковий текст програми у вікні екранного редактору комплексу *SCM* МК48, виконати пошук синтаксичних помилок та скомпілювати програму;

– налагодити розроблену програму множення за допомогою програмно-логічної моделі *SCM* МК48 з отриманням повної інформації про хід виконання програми.

2. 3. Зробити висновки.

Зміст звіту

Звіт повинен містити короткі теоретичні відомості, операційну схему, цифрову діаграму стану регістрів, схему алгоритму, налагоджену програму на символічному асемблері МК48, висновки за роботою.

Контрольні питання

1. Охарактеризуйте основні способи множення чисел.
2. Поясніть в яких випадках і як виконується округлення результату під час виконання арифметичних операції?
3. Розробіть алгоритм та підпрограму виконання наступних арифметичних операцій у МК48:
   – додавання,
   – віднімання,
   – зсув шістнадцятирозрядних слів,
   – якщо операнди знаходяться
   – у внутрішній пам'яті даних,
   – у зовнішній пам'яті даних.
4. Поясніть як здійснити перевірку значення довільного розряду регістра
   – у внутрішній пам'яті даних,
   – у зовнішній пам'яті даних.

7.1.5. Практична робота 2.5

ОБЧИСЛЕННЯ АРИФМЕТИЧНОЇ ФУНКЦІЇ В МІКРОКОНТРОЛЕРІ КМ1816ВЕ48

Мета роботи: Вивчення структури, режимів роботи, системи команд і отримання навиків розробки програм виконання широкого набору арифметичних операцій в мікроконтролері КМ1816ВЕ48.

Теоретичні відомості: 3.3, 3.4, 3.5.2, 3.5.4, [1], [15].

Підготовка до виконання практичної роботи

1. Записати номер варіанту (номер залікової книжки) у двійковому поданні і виділити шість молодших розрядів $h_6 - h_1$.

2. За отриманими значеннями двійкових розрядів вибрати спосіб ділення з табл. 7.13 та задану арифметичну функцію з табл. 7.14. Розрядність операндів прийняти рівній восьми. Машинні коди, в яких повинні бути подані операнди *X* та *Y* у вихідному стані, та результат Z наприкінці виконання обчислень, визначені в табл. 7.15, де ПК – прямий код, ДК – доповнювальний код.

3. Розробити операційну схему, цифрові діаграми стану регістрів, схему алгоритму обчислення заданої функції $F = (X1, X2)$ (де $X1 < 1, X2 < 1, X2 < X1, X1 \neq 0$). Значення операндів вибрати самостійно.

Таблиця 7.13. Варіанти завдання

h_1	**Спосіб ділення**
0	Ділення правильних дробів з відновленням залишку
1	Ділення правильних дробів без відновлення залишку

4. Розробити програму обчислення заданої функції на асемблері МК48. Порти для вводу/виводу даних, надані в табл. 7.16.

5. Визначити час виконання операції при опорній частоті F=6МГц.

Таблиця 7.14. Варіанти завдання

h_2	h_1	**Арифметична функція**
0	0	$F = X2 / X1 + 2(X2 + X1)$
0	1	$F = 4(X2 + X1) - X2 / X1$
1	0	$F = X2 / X1 + (X2 + X1) / 2$

1	1	$F = (X2 - X1)/4 + X2/X1$

Таблиця 7.15. Варіанти завдання

h_1	h_4	h_5	***X*1**	***X*2**	***F***
0	0	0	ДК	ДК	ПК
0	0	1	ПК	ПК	ДК
0	1	0	ПК	ДК	ПК
0	1	1	ДК	ПК	ПК
1	0	0	ПК	ПК	ДК
1	0	1	ДК	ДК	ПК
1	1	0	ДК	ПК	ДК
1	1	1	ПК	ДК	ДК

Таблиця 7.16. Варіанти завдань

h_2	h_4	**Порт**
0	0	*P*2
0	1	*P*4, *P*5
1	0	*BUS*
1	1	*P*1

Порядок виконання роботи

1. Використовуючи моделюючий комплекс *SCM* MK48 налагодити розроблену програму:

– сформувати початковий текст програми у вікні екранного редактору комплексу *SCM* MK48, виконати пошук синтаксичних помилок та скомпілювати програму;

– налагодити розроблену програму пристрою управління за допомогою програмно-логічної моделі *SCM* MK48 з отриманням повної інформації про хід виконання програми.

2. Зробити висновки.

Контрольні питання

1. Охарактеризуйте основні методи ділення чисел.

2. Поясніть в яких випадках і як виконується округлення результату під час виконання операції ділення?

3. Поясніть які обмеження накладаються на операнди під час ділення чисел з фіксованою комою?

4. Поясніть, як організувати цикл
   – з виходом із циклу за умовою,
   – з виходом із циклу за заданій кількості проходів.

5. Яким чином здійснити запис результату ділення в задану сторінку зовнішньої пам'яті даних.

6. Намалювати структурну схему підключення до МК48 п'яти сторінок ЗПД.

7.2. Задачі для самостійного розв'язування

Задача 7.1. Розробити для МК48 програму реалізації заданого алгоритму управління. Для вводу/виводу сигналів використати порт *P*2. Розряди *P*2[7] та *P*2[6] в початковому стані налагоджені на ввід інформації, а розряди *P*2[5] та *P*2[0] – на вивід інформації.

Вихідні дані:

- Алгоритми управління:

а). П $X_1 \overset{1}{\uparrow} Y_3 \overset{1}{\downarrow} Y_4 Y_5 X_2 \overset{2}{\uparrow} Y_3 \overset{2}{\downarrow}$ К

Прийняти: $t_{Y_5} \geq 100 мкс$; $t_{Y_4} \geq 65 мкс$, $t_{Y_3} \geq 28 мкс$.

б). П $(Y_1 Y_2) X_1 \overset{1}{\uparrow} (Y_3 Y_4 Y_5) X_2 \overset{2}{\uparrow} Y_3 \overset{2}{\downarrow} \overset{1}{\downarrow}$ К

Прийняти: $t_{Y_1} = t_{Y_2} \geq 50 мкс$; $t_{Y_5} \geq 600 мкс$, $t_{Y_3} = t_{Y_4} \geq 40 мкс$.

в). П $Y_4 X_1 \overset{1}{\uparrow} Y_2 Y_5 \overset{1}{\downarrow} Y_3$ К

Прийняти: $t_{Y_5} > 660 мкс$; $t_{Y_4} = t_{Y_2} > 40 мкс$, $t_{Y_3} = 25 мкс$.

Задача 7.2. Розробити для МК48 програму обчислення заданого виразу. Намалювати структурну схему підключення зовнішньої пам'яті даних.

Вихідні дані:

- Вираз для обчислення:

$F = 3(R0 - R3) + (R5 \,\&\, R6)/4$.

- Джерело даних для обчислення:

  Дані в регістри *R*0, *R*3, *R*5, *R*6 першого банку регістрів завантажуються з комірок п'ятої сторінки зовнішньої пам'яті даних з адресами *C*0*h*, *C*1*h*, *C*2*h*, *C*3*h*.

Задача 7.3. Розробити для МК48 алгоритм та програму обчислення заданого виразу за умови встановлення ознаки *F*0 (*F*1). Намалювати структурну схему підключення зовнішньої пам'яті даних.

Вихідні дані:

- Вираз для обчислення:

а). $R1 := 4(R0 - R5) + K$,

де $K = (R6 + 5)/4$, за умови $F0 = 1$,

$K = R1.R2 + R3.R4$, за умови $F0 = 0$;

б). $R7 = 4(R0.R4 + R5) + K$,

де $K = R6/4$, за умови $F1 = 1$,

$K = 3(R1 - R2)$, за умови $F1 = 0$;

- Місце розташування результату:

в). Результат записати в зовнішню пам'ять даних.

г). Результат записати в комірки третьої сторінки зовнішньої пам'яті даних з адресами 22*h*, 23*h*, 24*h*, 25*h*.

д). Результат записати в комірки другої сторінки зовнішньої пам'яті даних з адресами *B*5*h*, *B*6*h*, *B*7*h*, *B*8*h*.

Задача 7.4. Розробити для МК48 алгоритм та програму обчислення заданого виразу. Намалювати структурну схему підключення зовнішньої пам'яті даних.

Вихідні дані:

- Вираз для обчислення:

а). $F = 4(R1.R4 - R2.R6) + FFh - R3$;

б). $F = 4(R3.R4 - R6) + 50h$,

де *Rr.Rk* – шістнадцятирозрядні операнди: *Rr* – старший байт слова, *Rk* – молодший байт.

- Джерело даних для обчислення:

с). у вихідному стані операнди знаходяться в першому банку регістрів;

д). у вихідному стані операнди знаходиться в другому банку регістрів.

- Місце розташування результату:

е). результат обчислення записати в комірки зовнішньої пам'яті даних з адресами *A*5*h* та *A*6*h*;

ж). результат обчислення записати в комірки зовнішньої пам'яті даних з адресами 64*h* та 6*Ch*.

Задача 7.5. Переслати дані із області даних зовнішньої пам'яті програм МК48 у регістри загального призначення.

Вихідні дані:

Поточна адреса пам'яті програм – 400. Дані прочитати із комірок з адресами 2265 та 3175. Дані записати в регістри *R*4 та *R*2 першого банку регістрів.

Задача 7.6. Переслати дані із області даних зовнішньої пам'яті програм МК48 у регістри загального призначення.

Вихідні дані:

Поточна адреса пам'яті програм – 840. Дані прочитати із комірок 1285 та 2677. Дані записати в регістри *R*1 та *R*5 першого банку регістрів та переслати в порт *PA* та порт *PB* ППА 580ВВ55.

Задача 7.7. Підключити до МК48 зовнішню пам'ять даних об'ємом 1Кб і зовнішню пам'ять програм об'ємом 4Кб. Розробити програму передачі байта даних з комірки пам'яті програм з адресою 4015 в акумулятор. Поточна адреса лічильника команд – 1022.

Задача 7.8. Розробити для МК48 програму обчислення різниці двох шістнадцятирозрядних чисел. Намалювати структурну схему підключення ЗПД.

Вихідні дані:

- Джерело даних для обчислення:

а). Вихідні числа зберігаються в комірках четвертої сторінки зовнішньої пам'яті даних з адресами: перше число – 2*h*, 22*h* та друге – 4*h*, 25*h*.

б). Вихідні числа зберігаються в комірках другої сторінки зовнішньої пам'яті даних з адресами: перше число – *C*2*h*, *B*0*h* та друге – *C*4*h*, *B*2*h*.

Задача 7.9. Розробити для МК48 програму обчислення різниці двох тридцятидвохрозрядних чисел. Намалювати структурну схему підключення ЗПД.

Вихідні дані:

- Джерело даних для обчислення:

а). Вихідні числа зберігаються в комірках шостої сторінки зовнішньої пам'яті даних з адресами: перше число – 30*h*, 31*h*, 32*h*, 33*h* та друге – *A*0*h*, *A*1*h*, *A*2*h*, *A*3*h*.

б). Вихідні числа зберігаються в комірках восьмої сторінки зовнішньої пам'яті даних з адресами: перше число – 36*h*, 35*h*, 34*h*, 33*h* та друге – *D*6*h*, *D*7*h*, *D*8*h*, *D*9*h*.

Задача 7.10. Розробити для МК48 програму множення цілих чисел за заданим способом множення. Розробити алгоритм, операційну схему множення чисел та цифрову діаграму стану регістрів. Намалювати структурну схему підключення ЗПД.

Вихідні дані:

- Розрядність чисел:
  а). чотири розряди;
  б). вісім розрядів;
- Спосіб множення:
  в). перший спосіб;
  г). другий спосіб;
  д). третій спосіб;
  е). четвертий спосіб.
- Джерело даних для обчислення:

з). Операнди знаходяться в комірках *BDh* та *BEh* третьої сторінки зовнішньої пам'яті даних.

к). Операнди знаходяться в комірках *A*8*h* та *A*9*h* сьомої сторінки зовнішньої пам'яті даних.

л). Операнди знаходяться в комірках 61*h* та *C*0*h* першої сторінки зовнішньої пам'яті даних.

- Об’єм зовнішньої пам'яті даних:

м). 4Кб;

н). 8Кб.

Задача 7.11. Розробити для МК48 із зовнішньою пам'яттю даних схему підключення ППА 580ВВ55. Адреси портів включити у загальний адресний простір зовнішньої пам'яті даних. Розробити програму передачі перших шістнадцяти байт зовнішньої пам'яті даних в порт *PB* ППА.

Вихідні дані:

- Адреси портів:

  PA = 0*Ch*, *PB* = 0*Eh*, *PC* = 0*Dh*, Регістр УСРР = 0*Fh*.

Задача 7.12. Розробити структурну схему підключення до МК48 однієї сторінки пам'яті програм, регістру станів (РС) і регістру даних (РД) одного зовнішнього пристрою і ППА 580ВВ55. Під час виконання задачі використати загальний адресний простір зовнішньої пам'яті даних. Адреси портів вибрати самостійно.

Задача 7.13. Розробити структурну схему підключення до МК48 трьох сторінок пам’яті даних, однієї сторінки пам'яті програм, об’ємом 1Кб, регістру станів (РС) і регістру даних (РД) одного зовнішнього пристрою та ППА 580ВВ55. Під час виконання задачі використати адресний простір першої сторінки зовнішньої пам'яті даних.

Вихідні дані:

- Адреси портів:

  PA = *A*0*h*, *PB* = *A*1*h*, *PC* = *A*2*h*, Регістр УСРР = *A*3*h*,
  РД = *A*4*h*, РС = *A*5*h*.

Задача 7.14. Розробити структурну схему підключення до МК48 однієї сторінки пам'яті даних, регістру станів (РС) і регістру даних (РД) одного зовнішнього пристрою. Під час виконання задачі використати загальний адресний простір зовнішньої

пам'яті даних. Розробити програму пересилки даних з комірки пам'яті даних з адресою *AAh* в регістр даних зовнішнього пристрою.

Вихідні дані:

- Адреси портів:

  РД = *CCh*, РС = *DCh*.

Задача 7.15. Розробити структурну схему підключення до МК48 трьох сторінок зовнішньої пам'яті даних та програмованого периферійного адаптера 580ВВ55. Розробити підпрограму передачі інформації з порту *PC* в регістр *R*5 першого банку резидентної пам'яті даних. Розробити програму обчислення заданого виразу. Результат обчислення виразу видати в порт *PA*. Під час виконання задачі використати адресний простір другої сторінки зовнішньої пам'яті даних. Детально розробити схему селектора адреси.

Вихідні дані:

- Вираз для обчислення:

  а). $(R5-R3)\times 3+R6/4\rightarrow R5$

  б). $5\times(R4+R3/2)+R6-1\rightarrow R6$

  в). $8\times R6-(R4-R3)\times 3+1\rightarrow R4$

- Адреси портів:

  PA = *F*0*h*, *PB* = *F*1*h*, *PC* = *F*2*h*, Регістр УСРР = *F*3*h*.

Задача 7.16. Розробити структурну схему підключення до МК48 п’яти сторінок зовнішньої пам'яті даних та програмованого периферійного адаптера 580ВВ55. Адреси портів *PA*, *PB*, *PC* програмованого периферійного адаптера входять в адресний простір третьої сторінки зовнішньої пам'яті даних. Розробити програму пересилки масиву даних з семи слів у внутрішню пам'ять МК48 розпочинаючи з комірки 053*h*. Детально розробити схему селектора адреси.

Вихідні дані:

- Адреси портів:

PA = A0h, PB = A1h, PC = A2h, Регістр УСРР = *A3h.*

Задача 7.17. Розробити структурну схему підключення до МК48 зовнішньої пам'яті даних об'ємом 4Кб. Розробити програму пересилки масиву з двадцяти слів із п'ятої сторінки зовнішньої пам'яті даних, розпочинаючи з комірки за адресою *C*0*h*. Перші вісім слів переслати у перший банк регістрів, інші у резидентну пам'ять даних розпочинаючи з комірки за адресою 040*h*.

Задача 7.18. Розробити структурну схему підключення до МК48 зовнішньої пам'яті даних об'ємом 1Кб. Розробити програму пересилки масиву *M* з двадцяти слів з другої сторінки зовнішньої пам'яті даних, розпочинаючи з комірки за адресою *A*0*h*. Масив переслати у резидентну пам'ять даних, розпочинаючи з комірки за адресою 30. Обчислити значення заданого виразу.

Вихідні дані:

- Вираз для обчислення:

а). $F = 2(X1 + X2 - 1) - (X3 + X4)$;

б). $F = (X1 + X2 + X3)/4 + 4(X4 - 1)$

- Джерело даних для обчислення:

в). *X*1, *X*2, *X*3, *X*4 – останні байти масиву *M*(20), *M*(19), *M*(18), *M*(17);
г). *X*1, *X*2, *X*3, *X*4 – перші байти масиву *M*(1), *M*(2), *M*(3), *M*(4).

Задача 7.19. Розробити структурну схему підключення до МК48 ППА 580ВВ55, двох сторінок зовнішньої пам'яті даних та додаткових портів *P*4, *P*5, *P*7 (без застосування ІС 580ВР43). Розробити програму пересилки даних із регістрів *R*5, *R*6 в комірки першої сторінки зовнішньої пам'яті даних з адресами *C*0*h*, *C*1*h* відповідно.

Задача 7.20. Розробити структурну схему підключення до МК48 п'яти сторінок зовнішньої пам'яті даних та додаткових портів *P*4, *P*6 (без застосування ІС 580ВР43). Розробити програму пересилки масиву даних із першого банка регістрів в другу сторінку зовнішньої пам'яті даних. Перше слово масиву розмістити у комірці пам'яті за адресою 44*h*. Кількість елементів масиву дорівнює 8.

Задача 7.21. Розробити структурну схему підключення до МК48 ППА 580ВВ55 та десяти зовнішніх пристроїв. Адреси додаткових портів та портів зовнішніх пристроїв входять у адресний простір третьої сторінки зовнішньої пам'яті даних. Для адрес портів виділити область зовнішньої пам'яті даних з адресами *D0h – FFh*. Детально розробити схему селектора адреси.

Задача 7.22. Розробити модуль ОЗП, об'ємом 512 Кб, для МПС з загальною шиною адреси та даних, якщо загальний адресний простір системи складає 4Мб, максимальна ширина вибірки слів – 4 байти, зчитування даних здійснюється байтами і словами. Детально розробити структурну схему блоку управління зчитуванням слів різної довжини.

Задача 7.23. Розробити модуль ПЗП, об'ємом 256 Кб, для МПС з розділеними шинами адреси і даних, якщо загальний адресний простір системи складає 2Мб, максимальна ширина вибірки слів – 4 байти, зчитування даних здійснюється словами подвоєної довжини та байтами.

Задача 7.24. Розробити модуль ОЗП, об'ємом 128 Кб, для МПС з розділеними шинами адреси і даних, якщо загальний адресний простір системи складає 2Мб, максимальна ширина вибірки слів – 4 байти, зчитування даних здійснюється байтами, словами та словами подвоєної довжини. Детально розробити структурну схему блоку управління зчитуванням слів різної довжини.

Задача 7.25. Розробити модуль ОЗП, об’ємом 128Кб, для МПС з загальною шиною адреси та даних, якщо загальний адресний простір є 1М, максимальна ширина вибору даних – 4 байти, зчитування інформації здійснюється байтами. Побудувати часові діаграми циклів запису, читання та читання-модифікації запису для побудованої МПС.

Задача 7.26. Розробити модуль ОЗП, об’ємом 64Кб, для МПС з розділеними шинами адресу і даних, якщо загальний адресний простір є 2М, максимальна ширина вибірки – 2 байти, зчитування інформації здійснюється байтами. Побудувати часові діаграми циклів запису, читання та читання-модифікації запису для побудованої МПС.

Задача 7.27. Розробити схему підключення до МПС із загальною шиною адреси та даних двох портів для вводу та виводу даних відповідно, адреси портів включити у загальний адресний простір ОП. Адреси портів A(PC1)= 0*7AEH*; A(PC2)= 0*8FEH*. Адреси РС та РД відрізняються молодшим розрядом. Побудувати часові діаграми циклів вводу та виводу даних для побудованої МПС.

Задача 7.27. Розробити схему підключення до МПС із розділеною шиною адреси та даних двох портів для вводу даних. Адреси РС та РД відрізняються старшим розрядом. Побудувати часові діаграми циклів вводу даних для побудованої МПС.

Задача 7.28. Розробити схему підключення до МПС із розділеною шиною адреси та даних двох портів для вводу даних. Адреси РС та РД відрізняються старшим розрядом. Побудувати часові діаграми циклів вводу даних для побудованої МПС.

Задача 7.29. Розробити схему підключення до МПС із розділеною шиною адреси та даних зовнішнього пристрою. Побудувати часові діаграми циклів вводу та виводу даних для побудованої МПС. Написати програму полінгу вводу даних для розробленого зовнішнього пристрою.

8. МОДУЛЬ 3

ОДНОКРИСТАЛЬНИЙ МІКРОКОНТРОЛЕР КР1816ВЕ51

8.1. Лабораторний практикум

8.1.1. Практична робота 3.1

РОЗРОБКА ПРОГРАМ ПЕРЕДАЧІ ДАНИХ У МК51

Мета роботи: Вивчення системи команд, форматів подання даних та способів адресації операндів; вивчення команд передачі управління, команд пересилки даних та команд вибору банків регістрів; отримання навиків розробки програм на мові асемблеру МК51.

Теоретичні відомості: 4.1, 4.2.2, 4.2.3, 4.3.1, 4.3.2, 4.3.5, 4.3.7, [15].

Підготовка до виконання практичної роботи

1. Вивчити структуру МК51. Вивчити способи підключення сторінок зовнішньої пам'яті даних до МК51.

2. Розробити структурну схему підключення до МК51 заданої кількості сторінок пам'яті даних та пам'яті програм. Кількість сторінок обрати з табл. 8.1 та табл. 8.2.

3. Вивчити особливості виконання основних груп команд МК51.

4. Записати номер варіанту в двійковому поданні і виділити п'ять молодших розрядів $h_5 - h_1$. Вибрати арифметичну функцію для розробки програми за табл. 8.3.

Таблиця 8.1. Варіанти завдання

h_0	h_1	**Кількість сторінок ПП**	**Об'єм сторінки**
0	0	4	2 Кб
0	1	6	64 Кб
1	0	5	256 б
1	1	3	64 Кб

Таблиця 8.2. Варіанти завдання

h_0	h_1	**Кількість сторінок ПД**	**Об'єм сторінки**
0	0	10	256 б
0	1	6	64 Кб
1	0	18	2 Кб
1	1	9	64 Кб

5. Розробити алгоритм та програму на асемблері МК51 для пересилки масиву вихідних даних, що складається з заданої кількості слів (табл. 8.4) у резидентну пам'ять даних МК51 розпочинаючи з комірки за адресою заданою у табл. 8.5.

Таблиця 8.3. Варіанти завдання

h_2	h_1	Банк регістрів	Функція для обчислення
0	0	БР0	$F1 = 16(X_1 + X_2 - 1) \times (X_3 - X_4) - \frac{X_5 \,\&\, X_6}{8}$
0	1	БР1	$F2 = \frac{8(X_1 \vee X_2) \times (X_3 - X_4) - X_5 / X_6}{16}$
1	0	БР3	$F3 = \frac{16(X_1 - 1) \wedge X_2 - (X_3 + X_4) \times X_5}{2 \times X_6}$
1	1	БР4	$F4 = 8(X_1 - X_2 \times 4) + \frac{(X_3 \oplus X_4 - X_5) / X_6}{2}$

Таблиця 8.4. Варіанти завдання

h_3	h_4	Розмірність масиву
0	0	30
0	1	18
1	0	22
1	1	27

Таблиця 8.5. Варіанти завдання

h_3	h_4	h_2	Адреса початкової комірки пам'яті
0	0	0	20*h*
0	0	1	48*h*
0	1	0	28*h*
0	1	1	6*Fh*
1	0	0	52*h*
1	0	1	38*h*
1	1	0	4*Ah*
1	1	1	60*h*

6. Останні шість байтів масиву є вихідними аргументами для обчислення функції. Розробити алгоритм обчислення функції та

програму на асемблері МК48. Записати значення аргументів у заданий банк регістрів (табл. 8.3), в якому виконати обчислення функції. Результат обчислення розмісти у регістрах, заданих у табл. 8.6.

7. Виконати числовий приклад для значень аргументів, заданих у табл. 8.7.

Таблиця 8.6. Варіанти завдання

h_3	h_2	**Регістри**	**Банк регістрів**
0	0	*R*4.*R*3	БР2
0	1	*R*5.*R*6	БР4
1	0	*R*6.*R*7	БР0
1	1	*R*7.*R*6	БР1

Таблиця 8.7. Варіанти завдання

h_4	h_1	***X*1**	***X*2**	***X*3**	***X*4**	***X*5**	***X*6**	***X*7**	***X*8**	***X*9 … *X*30**
0	0	–7	12	–17	3	5	*FF*	*	*	* … *
0	1	12	*A*2	–11	*A*	–23	11	*	*	* … *
1	0	–18	-3	23	11	–7	*B*8	*	*	* … *
1	1	18	-3	*AA*	40	–10	18	*	*	* … *

Примітка: * – довільні дані.

Порядок виконання роботи

1. Використовуючи моделюючий комплекс *SCM* МК51 налагодити розроблену програму:

– сформувати початковий текст програми у вікні екранного редактору комплексу *SCM* МК51, виконати пошук синтаксичних помилок та скомпілювати програму;

– налагодити розроблену програму пристрою управління за допомогою програмно-логічної моделі *SCM* МК51 з отриманням повної інформації про хід виконання програми.

3. Зафіксувати в протоколі результат виконання тестового прикладу і часові діаграми сигналів, що забезпечують роботу із зовнішньою пам'яттю даних.

4. Зробити висновки.

Контрольні питання

1. Навести приклади команд роботи з бітами.

2. Пояснити де знаходиться область ознак. Яким чином очистити всі ознаки?

3. Пояснити під час виконання яких команд формується ознака *C*.

4. Яким чином здійснити переключення банків регістрів РПД?

5. Скільки таймерів/лічильників входять у склад МК51?

6. Навести приклади видавання на виводи порту *P*1[3..0] управляючих сигналів тривалістю:

а) 450 мкс;

б) 22 мкс;

в) 640мкс.

7. Розробити операційну схему и програму виконання зсувів тридцятидвохрозрядних слів:

а) на два розряди вліво;

б) на чотири розряди вправо;

в) на чотири розряди вліво.

8. Пояснити функціональне призначення портів вводу/виводу. Яким чином відбувається прийом інформаційного слова в МК51?

9. Розробити структурну схему підключення до МК51 додаткових портів вводу/виводу:

а) *P*5, *P*7;

б) *P*5, *P*6, *P*7;

в) *PA*, *PB*, *PC*.

10. Намалювати часову діаграму обміну даними між мікропроцесором та пам'яттю в режимі читання для МПС з розділеними шинами адреси та даних.

11. Намалювати часову діаграму обміну даними між мікропроцесором та пам'яттю в режимі читання-модифікація-запис для МПС з суміщеними шинами адреси та даних.

12. Який із наведених способів адресації операндів забезпечує мінімальній час виконання операції: пряма, непряма, безпосередня адресація? Поясніть чому?

13. Яке із наведених слів команди має найменшу довжину: з прямою, непрямою, автодекрементною адресацією?

8.1.2. Практична робота 3.2

ВИКОНАННЯ ОПЕРАЦІЇ ДОДАВАННЯ І ВІДНІМАННЯ З ПЛАВАЮЧОЮ КОМОЮ В МІКРОКОНТРОЛЕРІ КР 1816ВЕ51

Мета роботи: Вивчення структури пам'яті МК51, системи команд, форматів подання даних, та способів адресації операндів; отримання навиків розробки програм виконання простих арифметичних операцій над числами з плаваючою комою для МК КР1816ВЕ51.

Теоретичні відомості: 4.2, 4.3.1, 4.3.2, 4.3.7, [1], [15].

Додаткові теоретичні відомості

Додавання чисел із плаваючою комою

Суму двох чисел $X = 2^{P_X} M_X$ і $Y = 2^{P_Y} M_Y$, поданих у форматі із плаваючою комою, можна записати у вигляді

$$2^{P_X} M_X + 2^{P_Y} M_Y = 2^{P_Z} M_Z .$$

Для додавання чисел із плаваючою комою необхідно привести їх до загального порядку $P_{заг}$, в якості якого зручно обрати більший порядок з двох доданків $P_{заг} = \max(P_X, P_Y)$.

Під час цього зменшення за рахунок зсуву праворуч підлягає мантиса числа з меншим порядком. В противному випадку виникає переповнення розрядної сітки мантиси числа, що перетворюється. Після цього суму двох чисел можна подати у вигляді

$$2^{P_{заг}} M_X + 2^{P_{заг}} M'_Y = 2^{P_{заг}} (M_X + M'_Y),$$

де за M'_Y прийнято перетворену мантису числа з меншим порядком.

Виконання операцій додавання та віднімання чисел із плаваючою комою у загальному вигляді складається з наступних етапів: вирівнювання порядків; підсумовування мантис; визначення порядку результату; нормалізація результату; округлення результату; кінцева нормалізація результату.

Формат числа з плаваючою комою

Для реалізації арифметичних операцій з плаваючою комою у МК51 числа подаються у вигляді тридцятидвохрозрядного двійкового коду, де сім молодших розрядів старшого байту є порядком числа, а три молодші байти – мантисою числа. Старший

розряд старшого байту вказує на знак числа. Формат числа з плаваючою комою зображений на рис. 4.16 (див. 4.2.12).

Симетричний порядок подається в додатному коді і змінюється від (– 128) до 127, де старший розряд – знаковий. За зміщеного порядку використовується позитивне число без знаку від 0 до 255 (нульовому порядку відповідає зсув +126).

Підготовка до виконання практичної роботи

1. Вивчити архітектуру і систему команд МК51.
2. Вивчити способи виконання арифметичних операцій над числами з плаваючою комою.
3. Записати номер варіанту в двійковому поданні і виділити п'ять молодших розрядів $h_5 - h_1$. Вибрати арифметичну операцію для розробки програми за табл. 8.8. Кількість байт в мантисі визначається за табл. 8.8, двійковий код подання мантис за табл. 8.9, форма подання порядку – за табл. 8.10. Розміщення операндів і результату вибрати відповідно до табл. 8.11.

Таблиця 8.8. Варіанти завдання

h_1	**Операція**	**Довжина мантиси**
0	додавання	2 байта
1	віднімання	2 байта

Таблиця 8.9. Варіанти завдання

h_4	**Форма подання мантиси**
0	ПК
1	ДК

Таблиця 8.10. Варіанти завдання

h_2	**Форма подання порядку**
0	симетричний
1	зміщений

4. Розробити алгоритм виконання заданої операції, скласти програму його реалізації на асемблері МК51.
5. Визначити час виконання операції при опорній частоті генератора 6 МГц.

Таблиця 8.11. Варіанти завдання

h_5	**Перший операнд, результат**	**Другий операнд**
0	РПД	ЗПД
1	ЗПД	РПД

Порядок виконання роботи

2. Використовуючи моделюючий комплекс *SCM* MK51 налагодити розроблену програму:

– сформувати початковий текст програми у вікні екранного редактору комплексу *SCM* MK51, виконати пошук синтаксичних помилок та скомпілювати програму;

– налагодити розроблену програму пристрою управління за допомогою програмно-логічної моделі *SCM* MK51 з отриманням повної інформації про хід виконання програми.

2. Зафіксувати в протоколі результат виконання тестового прикладу і часові діаграми сигналів, що забезпечують роботу із зовнішньою пам'яттю даних.

3. Зробити висновки.

Зміст звіту

Звіт повинен містити короткі теоретичні відомості, схему алгоритму, налагоджену програму на асемблері, часові діаграми, а також висновки по роботі.

Контрольні питання

1. Посніть етапи виконання операцій з плаваючою комою.
2. Наведіть основні відмінності архітектури мікроконтролерів BE51 і BE48.
3. Назвіть способи адресації операндів в мікроконтролері BE51
4. Типи команд. Формат команд. Команди прямої і непрямої адресації.
5. Вкажіть призначення кожного з управляючих регістрів мікроконтролера.
6. Накресліть часову діаграму управляючих сигналів під час роботі MK51 із зовнішньою пам'яттю програм.
7. Як проаналізувати окремий розряд регістра спеціальних функцій?
8. У яких режимах можуть працювати таймери/лічильники? Охарактеризуйте кожен режим.

8.1.3. Практична робота 3.3

ВИКОНАННЯ СКЛАДНИХ АРИФМЕТИЧНИХ ОПЕРАЦІЙ З ПЛАВАЮЧОЮ КОМОЮ В МІКРОКОНТРОЛЕРІ КР 1816ВЕ51

Мета роботи: Вивчення структури МК51, системи команд і отримання навиків розробки програм виконання складних арифметичних операцій над числами з плаваючою комою для МК КР1816ВЕ51.

Теоретичні відомості: 4.2, 4.3.1, 4.3.2, 4.3.7, [1], [15].

Додаткові теоретичні відомості

Множення чисел із плаваючою комою

Множення двох чисел $X = 2^{P_X} M_X$ і $Y = 2^{P_Y} M_Y$, що задані у форматі із плаваючою комою, можна подати у вигляді

$$2^{P_Z} M_Z = 2^{P_X} M_X \cdot 2^{P_Y} M_Y = 2^{(P_X + P_Y)} \cdot (M_X \cdot M_Y).$$

Під час виконання операції множення мантис можна отримати результат із порушенням нормалізації вправо.

Можна виділити наступні *етапи множення чисел із плаваючою комою*: одержання порядку результату у вигляді суми порядків множників; знаходження мантиси результату, у вигляді добутку мантис множників; нормалізація результату, тобто приведення мантиси результату до вигляду $2^{-1} \le M_Z < 1$.

Ділення чисел із плаваючою комою

Результат ділення двох чисел $X = 2^{P_X} M_X$ і $Y = 2^{P_Y} M_Y$, що задані у форматі із плаваючою комою, можна записати у вигляді

$$2^{P_Z} M_Z = \frac{2^{P_X} M_X}{2^{P_Y} M_Y} = 2^{(P_X - P_Y)} \cdot \frac{M_X}{M_Y}.$$

Ділення мантис повинно виконуватись при виконанні умови $M_X < M_Y$, яка не завжди виконується при поданні мантис у нормалізованій формі. Тому перед началом ділення мантису діленого завжди зсувають вправо, чим забезпечують зменшення її у два рази. Відповідно до порядку діленого додається одиниця, тобто

$$2^{P_X} M_X = 2^{P_X + 1} \cdot M_X \cdot 2^{-1}.$$

Етапи ділення чисел із плаваючою комою наступні: одержання порядку результату із урахуванням виконання умови $M_X < M_Y$ шляхом віднімання порядку дільника від порядку діленого; одержання мантиси результату, на цьому етапі виконується ділення мантис; нормалізація результату. Нормалізація результату виконується аналогічно нормалізації під час множення чисел із плаваючою комою.

Добування квадратного кореня з числа із плаваючою комою

Добування квадратного кореня з числа $X = 2^{P_X} M_X$, що задане у форматі із плаваючою комою, можна подати у вигляді

$$Y = \sqrt{X} = \sqrt{2^{P_X} M_X} = 2^{P_X/2} \cdot \sqrt{M_X} .$$

Таким чином, для добування квадратного кореня з числа із плаваючою комою необхідно порядок числа поділити на два, а з мантиси добути квадратний корінь за правилами для чисел з фіксованою комою.

Ділення порядку на два відбувається шляхом зсуву його на один розряд вправо, якщо порядок парний. Якщо порядок непарний, то до нього необхідно додати одиницю, та зсунути мантису на один розряд вправо, після чого порядок зсувають на один розряд вправо – ділення на два. Тобто, якщо порядок $P_X = 2k–1$, то

$$X = 2^{2k-1} M_X = 2^{2k} (M_X \cdot 2^{-1}),\ Y = \sqrt{X} = 2^k \sqrt{M_X \cdot 2^{-1}} .$$

Мантиси чисел із плаваючою комою завжди нормалізовані і, коли у першому циклі з мантиси відбувається віднімання числа 0,01, перший залишок буде завжди додатним, таким чином перша цифра результату завжди буде дорівнювати одиниці. Відповідно то цього при виконанні операції добування квадратного кореня у пристрої із плаваючою комою не може відбутися порушення нормалізації.

Етапи добування квадратного кореню з числа із плаваючою комою наступні: одержання порядку результату; одержання мантиси результату, яка є завжди нормалізованою, тому етап нормалізації не виконується.

Перед початком операції знак операнда та сам операнд аналізується на рівність нулю. За нульовою мантисою добування квадратного корені не відбувається, а результату привласнюється значення нуля. Якщо знак операнда зворотній, то пристрій генерує сигнал помилки.

Підготовка до виконання практичної роботи

1. Вивчити архітектуру і систему команд МК51.

2. Вивчити способи виконання арифметичних операцій над числами з плаваючою комою.

3. Записати номер варіанту в двійковому поданні і виділити п'ять молодших розрядів $h_5 - h_1$. Вибрати арифметичну операцію для розробки програми за табл. 8.12. Кількість байт в мантисі визначається за табл. 8.12, двійковий код подання мантис за табл. 8.13, форма подання порядку – за табл. 8.14. Розташування операндів і результату вибрати відповідно до табл. 8.15.

Таблиця 8.12. Варіанти завдання

h_0	h_1	**Операція**	**Довжина мантиси**
0	0	множення	3 байта
0	1	ділення	3 байта
1	*	обчислення $\sqrt{X}$	3 байта

6. Розробити алгоритм виконання заданої операції, скласти програму його реалізації на асемблері МК51.

7. Визначити час виконання операції при опорній частоті генератора 6 МГц.

Таблиця 8.13. Варіанти завдання

h_4	**Форма подання мантиси**
0	ДК
1	ПК

Таблиця 8.14. Варіанти завдання

h_2	**Форма подання порядку**
0	зміщений
1	симетричний

Таблиця 8.15. Варіанти завдання

h_5	**Перший операнд, результат**	**Другий операнд**
0	ЗПД	РПД
1	РПД	ЗПД

Порядок виконання роботи

1. Використовуючи моделюючий комплекс *SCM* MK51 налагодити розроблену програму:

– сформувати початковий текст програми у вікні екранного редактору комплексу *SCM* MK51, виконати пошук синтаксичних помилок та скомпілювати програму;

– налагодити розроблену програму пристрою управління за допомогою програмно-логічної моделі *SCM* MK51 з отриманням повної інформації про хід виконання програми.

2. Зафіксувати в протоколі результат виконання тестового прикладу і часові діаграми сигналів, що забезпечують роботу із зовнішньою пам'яттю даних.

3. Зробити висновки.

Зміст звіту

Звіт повинен містити короткі теоретичні відомості, схему алгоритму, налагоджену програму на асемблері, часові діаграми, а також висновки за роботою.

Контрольні питання

1. Посніть етапи виконання операцій з плаваючою комою
2. Наведіть основні відмінності архітектури мікроконтролерів BE51 і BE48.
3. Назвіть способи адресації операндів в мікроконтролері BE51
4. Вкажіть призначення кожного з керуючих регістрів мікроконтролера.
5. У яких режимах можуть працювати таймери/лічильники? Охарактеризуйте кожен режим.
6. Накресліть часову діаграму керуючих сигналів при роботі мікроконтролера із зовнішньою пам'яттю програм.
7. Як проаналізувати окремий розряд регістра спеціальних функцій?

8.1.4. Практична робота 3.4

ВИВЧЕННЯ УЧБОВО-НАЛАГОДЖУВАЛЬНОГО СТЕНДУ

Мета роботи: Вивчення функціональних можливостей учбово-налагоджувального стенду, внутрішньої структури і системи

команд МК51; вивчення команд пересилок, арифметичних, логічних команд, команд переходів. Взаємодія внутрішніх вузлів МК51.

Теоретичні відомості: 4.2, 4.3.1, Додаток Г.

Додаткові теоретичні відомості

Опис структури та принципу функціонування навчально-налагоджувального стенду МК51 для виконання практичних робіт 3.4 – 3.6 наведений у додатку Г.

Робота з учбово-налагоджувальним стендом МК51

Для завантаження програми у учбово-налагоджуваний стенд МК51 (УНС) необхідно виконати наступну послідовність дій:

1. На персональному комп'ютері запустити текстовий редактор.
2. В текстовому редакторі набрати текст програми у мнемокодах асемблеру МК51 (або на мові програмування С).
3. Зберегти файл з розширенням *. *ASM* (*.С).
4. Скомпілювати програму відповідними засобами.
5. Можливі синтаксичні помилки можна продивитись у однойменному файлі з розширенням *. *LST*.
6. Після виправлення помилок дані, які збережені у файлі з розширенням *. *HEX* за допомогою програми *EVAL*32.*EXE* перенести у УНС МК51. Довідкова інформація про параметри програми *EVAL*32.*EXE* можна отримати під час запуску *EVAL*32.*EXE*.
7. Під час передачі даних з персонального комп'ютера у стенд на екрані монітору відображуються дані, що передаються. Ці ж дані відображуються на індикаторі стенду – світиться світлодіод *HL*9.
8. Зупинка завантаженої програми і перехід у режим очікування прийому даних з персонального комп'ютера здійснюється натисканням кнопки *SW*2. При цьому світлодіод *HL*9 виключається.
9. Запис нової програми можливо здійснити в любий момент часу роботи завантаженої програми.

Для формування початкового тексту програми у мнемокодах асемблеру МК51, виконання пошуку синтаксичних помилок, компілювання програми, налагодження та отримання файлу з розширенням *. *HEX* для завантаження у УНС МК51 можна використовувати моделюючий комплекс *SCM* МК51. Особливо це рекомендується робити для розробки складних програми виконання

арифметичних перетворювань для здійснення покрокового налагодження програми.

Розробка програми

Приклад 8.1. Вміст регістрів *R*1 і *R*4 скласти і відобразити на індикаторі

```
        ORG     0
        MOV     R1,#04h         ; записати в R1 число 04
        MOV     R4,#30h         ; встановити в R4 число 30
        MOV     DPTR,#0B000h    ; встановити в DPTR адресу
                                ; індикаторів DD17, DD18
        MOV     A,R1            ; записати в ACC значення R1
        ADD     A,R4            ; скласти значення A і R4
                                ; результат суми в A
        MOVX    @DPTR,A         ; засвітити на індикаторах
                                ; DD17, DD18 число, яке
                                ; зберігається в ACC
        MOV     A,#00h          ; завантажити нулі в  ACC
CON:    JMP     CON             ; перехід на зациклення
                                ; програми
        END
```

Підготовка до виконання практичної роботи

1. Вивчити структурну схему стенду, розподіл пам'яті, призначенні вузлів. Вивчити структуру МК51. Вивчити синтаксис команд пересилки, арифметичних команд, команд переходів.
2. Розробити алгоритм для виконання індивідуального завдання за табл. 8.16.
3. Розробити програму для виконання індивідуального завдання на асемблері МК51.
4. Вивчити програмно-налагоджувальні засоби для МК51.
5. Ввести програму індивідуального завдання у комп’ютер.

Таблиця 8.16. Варіанти індивідуального завдання

$h_2\ h_3\ h_1\ h_0$ [16]	**Завдання**
0	Занести в регістр *R*4 двійково-десяткове число 0*, в регістр *R*6 двійково-десяткове число *0, суму чисел

	відобразити на першому і другому знакомісці статичної індикації.
1	Занести в регістр *R*3 двійково-десяткове число **, відобразити його на першому і четвертому знакомісці статичної індикації.
2	Занести в регістр *B* двійково-десяткове число **, з частотою 2 Гц, виводити це число на першому і другому знакомісці статичної індикації.
3	Занести в акумулятор двійково-десяткове число **, в регістр *R*5 – X0, число з акумулятора відобразити на першому і другому знакомісці статичної індикації, число з *R*5 відобразити на третьому знакомісці статичної індикації.
4	Занести в регістр *R*2 двійково-десяткове число 0*, в регістр *R*5 – *0, суму чисел відобразити на другому і третьому знакомісці статичної індикації.
5	Занести в комірку з адресою *B*0*h* резидентної пам'яті даних двійково-десяткове число 0*, в регістр *R*3 число *0, суму чисел відобразити на другому і третьому знакомісці статичної індикації з частотою 0,5 Гц.
6	Занести в регістр *R*0 двійково-десяткове число **, поперемінно відображати молодшу і старшу тетраду на першому і четвертому знакомісці статичної індикації з частотою 1 Гц.
7	Занести у регістр *B* двійково-десяткове число *0, в регістр *R*1 – **, число з регістру *B* відображати на першому знакомісці статичної індикації з частотою 1 Гц, число з *R*1 відображати на третьому і четвертому знакомісці статичної індикації з частотою 0,5 Гц.
8	Підрахувати значення регістра *TCON* і відобразити його на третьому і четвертому знакомісці статичної індикації.
9	Занести в регістр *R*4 двійково-десяткове число 0*, в регістр *R*3 – *0, суму чисел відобразити на другому і третьому знакомісці статичної індикації з повільним (на протязі 5 сек.) загасанням цього числа.
A	Занести в акумулятор двійково-десяткове число *0, в регістр *B* – 0*, суму чисел відобразити на першому і четвертому знакомісці статичної індикації.

B	Занести в регістр *B* двійково-десяткове число 0*, в регістр *R*5 – *0, два розряди суми (десятки і одиниці) по черзі відображати на першому і другому знакомісці статичної індикації.
C	Занести в регістр *R*1 двійково-десяткове число 0*, віднімаючи від числа одиницю, відображати на третьому знакомісці статичної індикації, набутого значення до нуля з частотою 1Гц.
D	Занести в регістр *R*3 двійково-десяткове число **, в регістр *R*5 – **, поперемінно відображати ці числа на першому і другому знакомісці статичної індикації (*R*3) і на третьому і четвертому знакомісці статичної індикації (*R*5).
E	Занести в регістр *A* двійково-десяткове число 0*, в регістр *R*2 – *0, число з *A* відобразити на четвертому знакомісці статичної індикації, число з регістра *R*2 відображати на другому знакомісці статичної індикації з частотою 0,5Гц.
F	Занести в регістр *R*0 число **, вивести на лівій парі знакомісць статичної індикації. Через 2 секунди вивести число **, занесене в регістр *R*1. Через 2 секунди на правій парі знакомісць статичної індикації вивести різницю чисел занесених в регістр *R*0 і *R*1.

де * – довільне значення у тетроді від 0 до *Fh*

Порядок виконання практичної роботи

1. Використовуючи програмно-налагоджувальні засоби для МК51 або моделюючий комплекс *SCM* МК51 налагодити розроблену програму:

– сформувати початковий текст програми за допомогою текстового редактора, виконати пошук синтаксичних помилок та скомпілювати програму;

– налагодити розроблену програму.

2. Завантажити програму в УНС МК51. Переконатися в правильному виконанні індивідуального завдання.

3. Роздрукувати лістинг програми.

Зміст звіту

Звіт повинен містити короткі теоретичні відомості, схему алгоритму, налагоджену програму на асемблері, часові діаграми, а також висновки за роботою.

Контрольні питання:

1. Як визначити час виконання команд. Пояснити поняття такту, машинного циклу.
2. Охарактеризувати такі типи команд, як команди зсуву, арифметичні команди, логічні команди. До якого класу належіть дані команди?
3. Пояснити призначення регістру ознак, навести команди що викликають зміну регістра ознак.
4. Які команди застосовуються для роботи із стеком. Навести послідовність команд під час роботі із стеком.
5. Навести призначення внутрішніх вузлів МК51.
6. Навести призначення і принцип роботи з внутрішньою пам'яттю даних МК.
7. Навести особливості реалізації системи переривання МК51. В чому призначення портів МК.
8. Описати фізичні характеристики вихідних сигналів МК51.

8.1.5. Практична робота 3.5

СПОСОБИ ПОБУДОВИ СХЕМ ВІДОБРАЖЕННЯ ІНФОРМАЦІЇ

Мета роботи: Вивчення схем динамічної і статичної індикації; отримати навики розробки програм для МК51 для відображення цифрової інформації на пристроях динамічного і статичного типу, а також на одиничних індикаторах.

Теоретичні відомості: 4.2, 4.3.1, Додаток Г.

Додаткові теоретичні відомості

Системи відображення інформації

Простими приладами відображення інформації в цифрових пристроях є *світлодіоди* і *цифрові індикатори*.

У напівпровідникових світлодіодних використовується властивість $(p-n)$ переходу випромінювати світло у видимій частині спектру при проходженні через нього прямого струму

($I_{пр}$ = 5 – 20 мА, $U_{пр}$ = 2 – 3 В). Варіанти включення індикаторів показані на рис. 8.1.

Для відображення цифрової інформації найбільшого поширення набули семисегментні індикатори, в яких відображення цифри складається з семи лінійних світлодіодних сегментів, розташованих у вигляді цифри вісім.

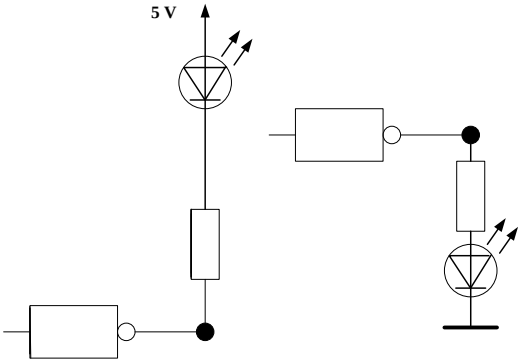


Рис. 8.1. Включення одиничних індикаторів.

На основі світлодіодів і семисегментних індикаторів будуються підсистеми відображення інформації, при побудові яких розрізняють *динамічну* і *статичну* схеми побудови підсистем індикації.

Статична індикація полягає в постійному підсвічуванні індикаторів *HL*1.0 – *HL*1.3 від одного джерела інформації рис. 8.2.

На структурній схемі статичної індикації (рис. 8.2) застосовується наступні позначення:

DA	–	дешифратор адреси, необхідний для вибірки відповідного регістра;
*R*1 – *R*4	–	регістри, в яких тимчасово зберігається значення коду числа для відображення (відповідний регістр вибирається *DA*);
*DC*1 – *DC*4	–	семисегментні дешифратори, що перетворюють двійковий код в семисегментний код;
*HL*1.0 – *HL*1.3	–	семисегментні індикатори;
ШД	–	шина даних, за якої здійснюється передача даних на індикацію.

У такій системі кожен з індикаторів *HL*1.*i* підключений через власний дешифратор *DCi* і регістр-клямку *Ri* до шини даних (де *i* – номер індикатора). Вибірка регістрів *Ri* здійснюється за допомогою селектора адреси (СА). Апаратні витрати при такій організації складають *n* пар регістрів та дешифратор для *n* десяткових розрядів індикатора (де *n* – кількість індикаторів).

Суть динамічної індикації полягає в почерговому циклічному підключенні кожного з індикаторів *HL*1.*i* до джерела інформації через загальну шину даних, рис. 8.3.

На структурній схемі динамічної індикації (рис. 8.3) застосовується наступні позначення:

DA	–	дешифратор адреси, для перетворення адреси що задається двійковим кодом в позиційний код;
RD	–	регістр даних для тимчасового зберігання числа, що відображається, або символу;
RA	–	регістр адреси для тимчасового зберігання двійкового коду адреси вибираного індикатора;
*HL*1.0 – *HL*1.3	–	семисегментні індикатори.

Вибірка індикатора здійснюється дешифратором адреси *DA*. У регістрі *RD* зберігається цифровий код, призначений для відображення. У регістрі *RA* зберігається адреса індикатора.

При такому включенні значно зменшуються апаратурні витрати. Але необхідно забезпечити достатній час для підсвічування одного індикатора, щоб не зменшувалася яскравість. Також необхідно забезпечити таку частоту перебору індикаторів, щоб не було помітно мерехтіння. Переваги такого способу помітні при кількості розрядів індикації більше п’яти.

В УНС МК51 статична індикація реалізована чотирма статичними семисегментними індикаторами *HG*1 (розряди *HG*1.0, *HG*1.1, *HG*1.2, *HG*1.3). Адреси індикаторів входять у загальний адресний простір пам’яті. Звернення до них здійснюється, як до елементів пам'яті з адресами *A*000*h* (ліва пара знакомісць) та *B*000*h* (права пара знакомісць).

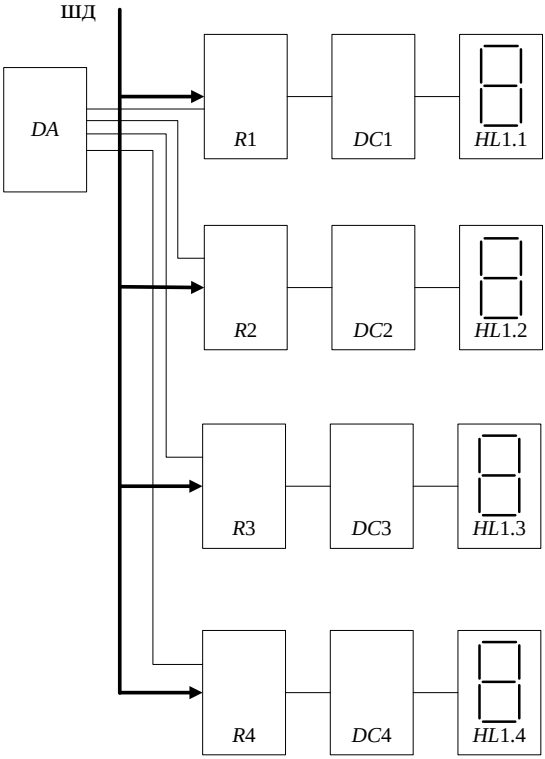


Рис. 8.2. Структурна схема статичної індикації.

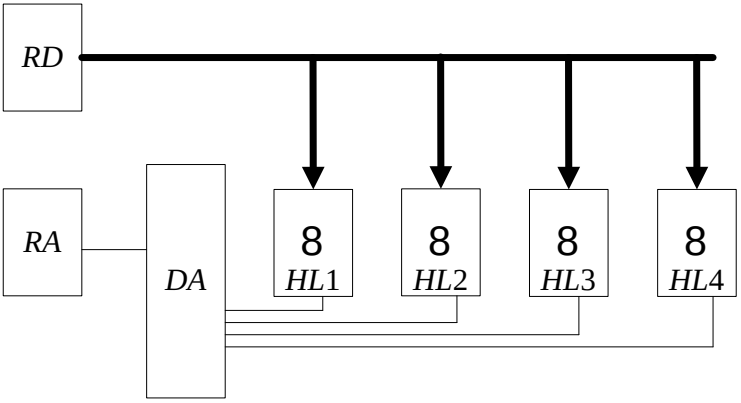


Рис. 8.3. Структурна схема динамічної індикації.

Динамічна індикація в УНС МК51 реалізована на платі розширення за допомогою чотирирозрядного семисегментного індикатора *HL*2. Управління динамічною індикацією здійснюється за допомогою порту *B* мікросхеми системного контролера (див. структурну схему УНС додаток Г), сигнали вибірки відповідного індикатора поступають від лінії порту *PC*0, *PC*1 до дешифратора адреси розряду *DD*3.

Знакосинтезуюча індикація в УНС МК51 реалізована на платі розширення за допомогою матриці світлодіодів 5×7 – *HG*1. Управління матрицею світлодіодів здійснюється за лініями *PA*0 – *PA*4 і *PC*0 – *PC*6. Наприклад, для того щоб засвітити крапку з координатами [1; 1] необхідно встановити рівень логічної одиниці на лінії *PA*0, і рівень логічного нуля на лінії *PC*0.

Також в УНС МК51 є лінійка світлодіодів *HL*1 – *HL*8 доступ до яких здійснюється як до комірки зовнішнього ОЗП за адресою 0*A*006*h*. Світлодіоди засвічуються шляхом запису логічних одиниць у відповідні розряди.

Принципова схема для виконання практичної роботи зображена на рис. 8.4.

Далі наведені приклади розробки програм для відображення цифрової інформації на пристроях динамічного і статичного типу для УНС МК51.

Приклад 8.2. Статична індикація. З частотою 1 Гц відобразити на статичному індикаторі число 04

```
          CSEG
          ORG   0
CONTINUE:
          MOV   A,#0
          MOV   DPTR,#0A004h
          MOVX  @DPTR,A          ; відмінити гасіння
                                 ; знакомісць С_інд
          MOV   A,#04h           ; записати в ACC число 04
          MOV   DPTR,#0A000h     ; встановити в DPTR
                                 ; адресу лівої пари
                                 ; знакомісць С_інд
          MOV   @DPTR,A          ; засвітити число 04
```

```
            MOV   DPTR,#0B000h   ; встановити в DPTR
                                 ; адресу правої пари
                                 ; знакомісць C_інд
            MOVX  @DPTR,A        ; засвітити число 04
            CALL  ZAD            ; виклик підпрограми
                                 ; затримки
            MOV   A,#00001111b
            MOV   DPTR,#0A004h
            MOVX  @DPTR,A        ; погасити всі знакомісця
                                 ; C_інд
            CALL  ZAD
            JMP   Continue       ; перехід на початок
                                 ; програми
ZAD:        MOV   R1,#0FFH       ; підпрограма затримки
C2:         MOV   R2,#0ffh
C4:         DJNZ  R2,C4
            DJNZ  R1,C2
            RET                  ; вихід з підпрограми
            END
```

Приклад 8.3. На світлодіодах *HL*1-*HL*8 запустити « одиницю, що біжить » зменшуючи час затримки між засвічуванням світлодіодів до певного значення, після чого засвітити всі світлодіоди (програма написана на мові програмування *C*).

```
#include <8051h>
#include "..\ev8031.lib\ev8031.c"
#include "..\ev8031.lib\bitdef.h"

int main ()
{
  unsigned int svet, low, i, zad;

  right i=0x33; // засвітити числа 33 на правому
  left i=0x33; // і лівому знакомісцях індикатора HG1
  zad=10000;
```

```
begin:
  low=1;
  LED_REG=low; // у регістр світлодіодів записуємо значення low
  for (svet=1; svet<8; svet++) // номер світлодіода
                                // HL1 – HL8, що засвітився
  {
    delay16 (zad);
    low=low<<1; // зсунути значення low на один знак вліво
    LED_REG=low;
  }
  low=256;
  for (svet=1; svet<8; svet++)
  {
    zad=zad-100;
    if (zad==100) // зменшити затримку
    goto cont; // якщо змінна zad досягла значення 100,
               // засвітити всі світлодіоди
    else
    delay16 (zad);
    low=low>>1; // зсунути значення low на один знак вправо
    LED_REG=low;
  }
  goto begin;
cont:
  low=0xFF;
  LED_REG=low;
  for (i=1; i<=100; i++)
  delay16 (1000);
  zad=10000;
  goto begin;
}
```

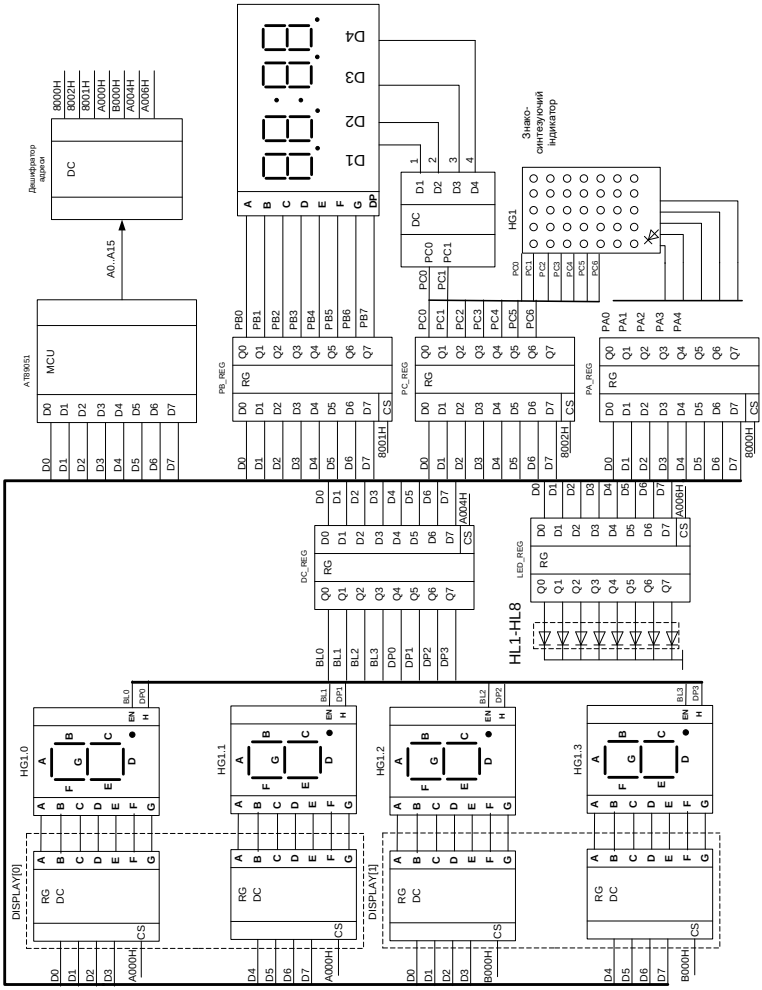


Рис. 8.4. Принципова схема до виконання практичної роботи

3.4. Підготовка до виконання практичної роботи

1. Вивчити принцип роботи різних методів відображення.
2. Розробити алгоритм для виконання індивідуального завдання.
3. Розробити програму для виконання індивідуального завдання (табл. 8.17).
4. Вивчити ПНЗ для КР1816ВЕ51.

Таблиця 8.17. Варіанти індивідуального завдання

h_2 h_4 h_6 h_0 [16]	**Завдання**
1	Занести в регістр *R*1 двійково-десяткове число **, віднімаючи від числа одиницю, відображати результат на динамічному індикаторі в молодшому розряді до нуля з частотою 0,5 Гц. Включати світлодіод, що біжить, на *HL*1 – *HL*8.
2	Занести у В двійково-десяткове число *0, в регістр *R*1 – **, число з В відобразити на знакосинтезуючому індикаторі, число з *R*1 відображати на динамічному індикаторі в старшому розряді з частотою 0,5 Гц.
3	Включити в шаховому порядку світлодіоди *HL*1 – *HL*8. Занести в регістр двійково-десяткове число 0*, в регістр *R*5 – *0, два розряди суми (десятки і одиниці) по черзі відображати на статичному індикаторі і на динамічному індикаторі з частотою 1Гц.
4	Занести в *R*6 двійково-десяткове число **, в *R*5 двійково-десяткове **, в *R*0 двійково-десяткове число **, відображати ці числа з *R*5, *R*6 на динамічному індикаторі, з *R*0 на статичному індикаторі.
5	Поперемінно відображати на знакосинтезуючому індикаторі число від 0 до 9, дублювати ці числа на динамічному індикаторі.
6	Занести в регістр А двійково-десяткове число 0*, в регістр *R*2 – *0, число з А відобразити на статичному індикаторі, число з регістра *R*2 відображати на динамічному індикаторі з частотою 0,6 Гц.
7	Занести в регістр А двійково-десяткове число **, в регістр R1 – **, молодші два розряди суми чисел

	відобразити на динамічному індикаторі, при цьому на знакосинтезуючому індикаторі здійснити плавне загорання числа 5.
8	Занести в регістр R6 число **, перетворити його в двійково-десяткове число, відобразити його на динамічному індикаторі, відобразити значення регістра *R*6 на світлодіодах *HL*1 – *HL*8 і його інверсний стан з частотою 1Гц.
9	Занести в регістр B двійково-десяткове число **, в регістр R3 – ** різницю чисел відобразити на динамічному індикаторі.
A	Відобразити на знакосинтезуючому індикаторі букву У. Занести в акумулятор число **, в регістр R5 – *0, число з акумулятора відобразити на статичному індикаторі, число з *R*5 відобразити на динамічному індикаторі.
B	Занести в регістр *R*0 двійково-десяткове число **, поперемінно відображати молодший і старший розряди на динамічному індикаторі з частотою 0,5 Гц.
C	Занести в регістр *R*2 двійково-десяткове число **, в регістр *R*5 **, суму відобразити на динамічному індикаторі.
D	Занести в регістр B двійково-десяткове число, з частотою 2 Гц виводити це число на статичному індикаторі і одночасно на динамічному індикаторі.
E	Поперемінно включати світлодіоди *HL*1 – *HL*8. Занести в комірку пам’яті з адресою 0010*h* зовнішньої пам'яті даних двійково-десяткове число 0*, в регістр R3 – **, суму чисел відобразити в старшому розряді на динамічному індикаторі.
F	Занести в регістр *R*1 двійково-десяткове число 0*, в регістр *R*3 **, суму відобразити на динамічному індикаторі. Шістнадцятирічне число відобразити на *HL*1 – *HL*8.
–	По черзі засвічуючи світлодіоди *HL*1 – *HL*8, на статичному індикаторі паралельно висвічувати кількість світлодіодів, що горять. Інтервал між засвічуванням 1с.
–	Відобразити на динамічному індикаторі слово з

	чотирьох букв. Вивести його таким чином: Перша буква, з крайнього правого положення пройшовши всі сегменти, залишається горіти в крайньому лівому положенні. Подальші букви, аналогічно, пройшовши всі сегменти, залишаються за попередніми. Коли слово повністю засвітилося, гасити сегменти по черзі, починаючи з крайнього лівого сегменту з інтервалом 0,5с.

де * – довільне значення у тетроді від 0 до *Fh*

Порядок виконання практичної роботи

1. Використовуючи програмно-налагоджувальні засоби:

– сформувати початковий текст програми за допомогою текстового редактора, виконати пошук синтаксичних помилок та скомпілювати програму;

– налагодити розроблену програму. За допомогою ПНЗ проаналізувати виконання індивідуальної програми.

2. Завантажити програму в УНС МК51. Переконатися в правильному виконанні індивідуального завдання.

3. Роздрукувати лістинг програми.

Зміст звіту

Звіт повинен містити короткі теоретичні відомості, схему алгоритму, налагоджену програму на асемблері, часові діаграми, а також висновки за роботою.

Контрольні питання

1. В чому особливості статичної та динамічної підсистеми відображення інформації? Як на УНС стенді реалізована статична та динамічна індикація?
2. Як провести розрахунок часу регенерації для динамічного методу відображення?
3. Обґрунтувати необхідності застосування різних методів відображення.
4. Привести схематичні рішення для побудови схем відображення інформації.
5. Привести схеми включення одиничних індикаторів.
6. Привести схеми включення рідкокристалічних індикаторів.
7. Привести схеми включення газорозрядних індикаторів.

8.1.6. Практична робота 3.6

СИСТЕМА ПЕРЕРИВАНЬ. ОПИТУВАННЯ ДИСКРЕТНИХ ДАТЧИКІВ

Мета роботи: Вивчення режимів роботи системи переривання МК, програмна обробка дискретних сигналів. Вивчення систем переривання режимів введення дискретної інформації, розробка програм опитування сигналів від датчиків.

Теоретичні відомості: 4.2, 4.2.7, Додаток Г.

Додаткові теоретичні відомості

Опитування дискретних сигналів

Для введення інформації широко застосовуються кнопкові перемикачі і контактні клавіатури. Сигнал таких перемикачів формується шляхом замикання (розмикання) електричного ланцюга. Сигнал, що формується контактною парою, супроводжується брязкотом, тривалість якого складає ~ 8-12мс рис. 8.5.

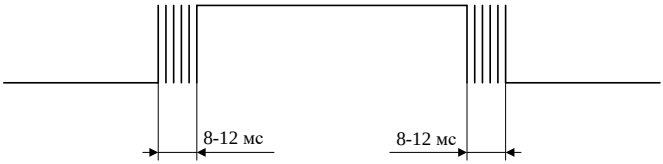


Рис. 8.5. Сигнал контактної пари

Для усунення брязкоту в отримуваному сигналі на виході контактної пари встановлюється спеціальні формувачі. Приклад такого формувача, заснованого на принципі безпосередньої установки *RS*-тригера, приведений на рис. 8.6.

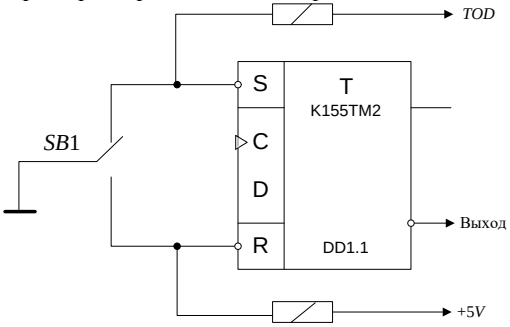


Рис. 8.6. Схема усунення брязкоту за допомогою *RS*-тригера

Для зменшення апаратних витрат застосовують програмне приглушення брязкоту. Воно полягає в повторному опитуванні контактної пари із затримкою в 12 мс, при збігу результатів опитування кнопка була натиснута, інакше в результаті першого опитування був зафіксований брязкіт.

Система переривань МК 1816ВЕ51

Регістр пріоритетів (*IP*) призначений для установки рівня пріоритету переривання для кожного з 5-ти джерел переривання.

Позначення розрядів регістра *IP* показане в табл. 8.18, а призначення вказане нижче.

Таблиця 8.18 Регістр пріоритетів переривання *IP*

7	6	5	4	3	2	1	0
X	*X*	*X*	*PSP*	*PT*1	*PX*1	*PT*0	*PX*0

Призначення розрядів регістра *IP* наступне:

*PX*0 – установка рівня пріоритету переривання від зовнішнього джерела /*INT*0;

*PT*0 – установка рівня пріоритету переривання від Т/Л0 (Т/Л – таймер/лічильник);

*PX*1 – установка рівня пріоритету переривання від зовнішнього джерела. /*INT*1;

*PT*1 – установка рівня пріоритету переривання від Т/Л1;

PS – установка рівня пріоритету переривання від послідовного порту;

X – резервний розряд.

Наявність в розряді *IP* одиниці встановлює для відповідного джерела високий рівень пріоритету, а наявність в розряді *IP* нуля – низький рівень пріоритету. Під час читання резервних розрядів, відповідні лінії магістралі даних не визначені.

Регістр дозволу переривання (*IE*) призначений для дозволу або заборони переривань від відповідних джерел. Позначення розрядів регістра *IE* показане в табл. 8.19.

Таблиця 8.19. Регістр дозволу переривань *IE*

7	6	5	4	3	2	1	0
EA	*X*	*X*	*ES*	*ET*1	*EX*1	*ET*0	*EX*0

Призначення розрядів регістра *IE* наступне:

EA	–	управління всіма джерелами переривань одночасно: *EA* = 0 – переривання заборонені, *EA* = 1 – переривання дозволені індивідуальними дозволами *EX*0, *ET*0, *EX*1, *ET*1, *ES*;
X	–	резервний розряд;
ES	–	управління перериванням від послідовного порту: *ES* = 1 – дозвіл, *ES* = 0 – заборона;
*ET*1	–	управління перериванням від Т/Л1: *ET*1=1 – дозвіл, *ET*1 = 0 – заборона;
*EX*1	–	управління перериванням від зовнішнього джерела /*INT*1: *EX*1 = 1 – дозвіл, *EX*1 = 0 – заборона;
*ET*0	–	управління перериванням від Т/Л0: *ET*0 = 1 – дозвіл, *ET*0 = 0 – заборона;
*EX*0	–	управління перериванням від зовнішнього джерела /*INT*0: *EX*0 = 1 – дозвіл, *EX*0 = 0 – заборона.

Під час читання резервних розрядів відповідні лінії магістралі не визначені.

Структура переривань

Механізм переривань в МК дозволяє автоматично реагувати на зовнішні і на внутрішні події (переповнювання таймерів/лічильників; завершення послідовного обміну).

Кожне із зовнішніх переривань /*INT*0, /*INT*1 може бути активізоване за рівнем або за фронтом сигналів *P*3.2, *P*3.3 за допомогою бітів *IT*0 і *IT*1 регістра *TCON*. Під час надходження запиту зовнішнього переривання /*INTx* встановлюється ознака *IEx* регістра *TCON*. Очищення ознаки *IEx* відбувається апаратно: під час переривання за фронтом *IEx* – скидається під час звернення до відповідної підпрограми обробки переривання; під час переривання за рівнем ознаки очищується за зняття запиту зовнішнього переривання, тобто в *IEx* відстежується стан виводу /*INTx*.

Щоб зовнішнє переривання за рівнем було розпізнане, необхідне, щоб низький рівень на виводі *INTx* утримувався на протязі не менш ніж дванадцять періодів сигналу тактової частоти. Якщо ж переривання активізується за переходом із стану високого рівня в стан низького рівня, то циклу низького рівня повинен передувати цикл високого рівня на виводі /*INTx*. Якщо зовнішнє переривання активізується за рівнем, запит повинен утримуватися до початку підпрограми обслуговування переривання і зніматися

перед завершенням цієї підпрограми для запобігання повторного обслуговуванню.

Переривання від таймерів/лічильників виконуються за ознаками *TF*0 і *TF*1 регістра *TCON*, які встановлюються під час переповнюванні відповідних регістрів таймерів/лічильників (за винятком режиму 3). Очищення ознак *TF*0 і *TF*1 відбувається під час переходу до підпрограми обслуговування переривання.

Переривання від послідовного порту виконується за ознакою закінчення прийому *R*1 або за ознакою закінчення передачі *T*1, які встановлюються в регістрі *SCON*.

На відміну від решти ознак, *R*1 і *T*1 скидаються тільки програмним шляхом зазвичай в межах підпрограми обробки переривання, де визначається, якому з прапорів *R*1 і *T*1 відповідає переривання.

У разі одночасного надходження запитів переривання з однаковим рівнем пріоритету, що дорівнює 0 або 1, обробка їх проводиться в порядку внутрішнього опитування ознак:

$$IE0 \to TF0 \to TE1 \to TF1 \to (T1 + R1).$$

Встановлення ознак переривання відбувається в кінці машинного циклу, а їх опитування в наступному циклі. І лише після виконання останнього циклу поточної команди відбувається апаратний виклик відповідної підпрограми обслуговування, еквівалентній команді *LCALL*.

У загальному випадку, звернення до підпрограми обслуговування переривання затримується під час виконання хоча б однієї з наступних умов:

– відбувається обробка переривання з таким самим або вищим пріоритетом.

– поточний машинний цикл (цикл опитування ознаки) не є останнім циклом команди, що виконується.

– виконується команда поточної програми *RET*1 або будь-яка команда звернення до регістрів *IE, IP*.

У останній умові після закінчення однієї з зазначених команд обов'язково виконається ще одна команда поточної програми перед викликом підпрограми обслуговування переривання.

Ознака переривання, що встановлена під час дії блокування переривання за однією з трьох вказаних умов і скинутих до їх зняття, не викличе обслуговування відповідного запиту переривання.

Підпрограма обслуговування переривання триває до виконання команди *RET*1, за якої відновлюється стан логіки переривання і стан програмного лічильника *PC* з двох верхніх комірок стеку. Під час використання команди *RET* відновлюється тільки стан програмного лічильника, а стан логіки переривання залишається незмінним.

Таблиця 8.20. Початкові адреси векторів переривань

Джерело переривання	**Адреса**
Зовнішнє переривання 0	0003*h*
Переповнювання таймера 0	000*Bh*
Зовнішнє переривання 1	0013*h*
Переповнювання таймера 1	001*Bh*
Послідовний порт	0023*h*

У складі УНС є дві окремі кнопки *S*10, *S*11, які можуть обпитуватися, як програмно, так і за допомогою використання функцій переривань *INT*0, *INT*1 відповідно.

У складі УНС є матрична 3×4 клавіатура *SW*3 – *SW*14 (табл. 8.21). Клавіатура підключена до шини даних МК за допомогою мікросхеми буфера *DD*1 74245 (АП6).

Опит всієї клавіатури відбувається за три рази (за один раз прочитується стан тільки одного стовпця клавіатури). Щоб провести опит стовпця клавіатури (*SW*3, *SW*6, *SW*9, *SW*12 або *SW*4, *SW*7, *SW*10, *SW*13 або *SW*5, *SW*8, *SW*11, *SW*14) необхідно виставити на відповідній лінії адреси (*A*0, *A*1, *A*2 для першого, другого і третього стовпця відповідно) рівень логічного нуля, а на інших лініях рівень логічної одиниці і прочитати стан буфера клавіатури, підключеного до шини даних МК, як доступний для читання елемент пам'яті з адресою 9000*h*. Якщо кнопка натиснута, то відповідний біт в байті буде рівний нулю, якщо ж не натиснута одиниця.

Таблиця 8.21. Робота із матричною клавіатурою

Стовпець (кнопки)	**Адреса**
1(*SW*3, *SW*6, *SW*9, *SW*12)	9006*h*
2(*SW*4, *SW*7, *SW*10, *SW*13)	9005*h*
3(*SW*5, *SW*8, *SW*11, *SW*14)	9003*h*

Принципова схема для виконання практичної роботи зображена на рис. 8.7.

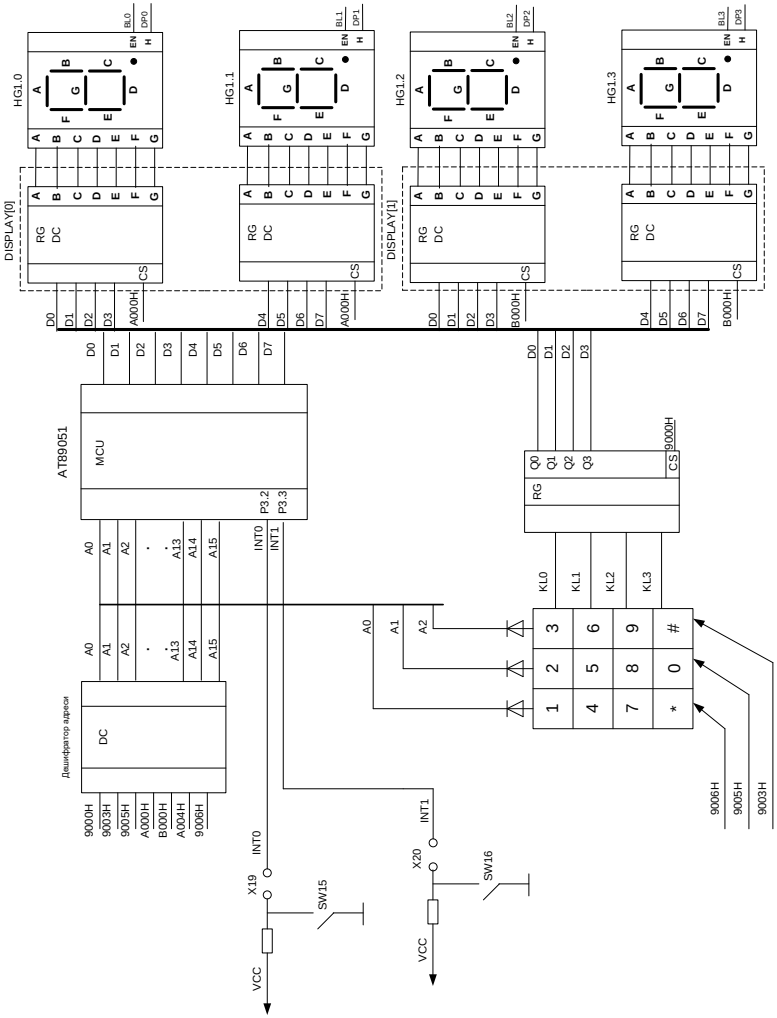


Рис. 8.7. Принципова схема до виконання практичної роботи 3.5

Далі наведені приклади розробки програм обробки дискретних сигналів для УНС.

Приклад 8.4. Приклад програмного опитування дискретного сигналу

```
          CSEG
          ORG    0
          JB     p3.2,$          ;опитування натискання кнопки
                                 ;SW15, якщо кнопка натиснута,
                                 ;програма виконується далі
CONTINUE:
          MOV    A,#0
          MOV    DPTR,#0A004h
          MOVX   @DPTR,A         ;відміна гасіння знакомісць C_інд
          MOV    A,#04h          ;записати в акумулятор число 04
          MOV    DPTR,#0A000h    ;встановити в DPTR адресу лівої
                                 ;пари знакомісць C_інд
          MOV    @DPTR,A         ;засвітити число 04
          MOV    DPTR,#0B000h    ;встановити в DPTR адресу правої
                                 пари знакомісць C_інд
          MOVX   @DPTR,A         ;засвітити число 04
          CALL   ZAD             ;виклик підпрограми затримки
          MOV    A,#00001111b
          MOV    DPTR,#0A004h
          MOVX   @DPTR,A         ;погасити всі знакомісця C_інд
          CALL   ZAD
          JMP    CONTINUE        ;перехід на початок програми
ZAD:                             ;підпрограма затримки
          MOV    R1,#0ffh
C2:       MOV    R2,#0ffh
C4:       DJNZ   R2,C4
          DJNZ   R1,C2
          RET                    ;вихід з підпрограми
          END
```

Приклад 8.5. Відображати на *C*_інд. Числа 33.33 і 32.32 при натисканні кнопок *SW*16, *SW*15 відповідно (програма написана на мові програмування С).

```
#include <8051h>
#include "..\ev8031.lib\ev8031.c"
#include "..\ev8031.lib\bitdef.h"

int main ()
{
begin:
   DC_REG=0x0F;  // гасити індикатор HG1
   if (!P3_2)
   {
     new_dotsi=0x20;
     left i=0x32;
     right i=0x32;  // якщо кнопка SW15 натиснута,
                    // вивести 32.32
   }
   if (!P3_3)
   {
     new_dotsi=0x20;
     left i=0x33;
     right i=0x33;  // якщо кнопка SW16 натиснута,
                    // вивести 33.33
   }
   goto begin;
}
```

Підготовка до виконання практичної роботи

1. Вивчити систему переривання КР1816ВЕ51, особливості опитування дискретних датчиків з механічними контактами.

2. Розробити алгоритм для виконання індивідуального завдання.

3. Розробити програму для виконання індивідуального завдання (табл. 8.22).

Таблиця 8.22. Варіанти індивідуальних завдань

$h_6\ h_2\ h_3\ h_0$ [16]	**Завдання**
1	Підрахувати і відобразити на статичному індикаторі кількість натиснень кнопки *SW*15.
2	Реалізувати опитування клавіатури. Номер клавіші

	відображати шляхом засвічування відповідної точки на знакосинтезуючому індикаторі.
3	Реалізувати опитування клавіатури. Номер клавіші послідовно відображати в кожному розряді динамічного індикатора.
4	Після натиснення *SW*15 запускати вогник, що біжить, на світлодіодах *HL*1 – *HL*8, під час опускання кнопки здійснюється плавне загорання числа 3 на знакосинтезуючому індикаторі.
5	Після натиснення кнопки *SW*16 включити секундомір з відображенням на статичному індикаторі значення секунд, під час опускання запускати «тінь», що біжить, на світлодіодах *HL*1 – *HL*8.
6	Реалізувати опитування клавіатури. Номер клавіші відображати позиційним кодом на світлодіодах *HL*1 – *HL*8, з відображенням кнопки на індикаторі.
7	Реалізувати програму введення чотиризначного числа з клавіатури, використовуючи статичний індикатор і дублюючи значення натиснутої кнопки на знакосинтезуючому індикаторі.
8	Реалізувати опит клавіатури після двох натискань кнопки *SW*16. Номер клавіші відображати на динамічному індикаторі.
9	За натиснення *SW*15 запускати будь-яке значення, що біжить, на знакосинтезуючому індикаторі, а за натиснення *SW*16 запалити всі точки в шаховому порядку.
A	Відображати значення секунд на статичному індикаторі. За переривання *INT*0 зупинити секундомір і засвітити світлодіоди *HLi* (*i* – непарне).
B	Відображати число 7543 на динамічному індикаторі. За переривання *INT*1 засвітити світлодіоди *HLi* (*i* – парне).
C	На статичному індикаторі відобразити число 5555. За переривання *INT*0 відображати «шахматку» на знакосинтезуючому індикаторі, за переривання *INT*1 відобразити на статичному індикаторі число 3333.
D	Після натиснення *SW*15 реалізувати програму введення тризначного числа з клавіатури з відображенням на статичному індикаторі.

E	Після натиснення *SW*16 запускати «тінь», що біжить, на знакосинтезуючому індикаторі, а за повторного натиснення *SW*16 згасити всі крапки.
F	Реалізувати опитування клавіатури. Номер клавіші відобразити двійковим кодом на світлодіодах *HL*1 – *HL*8.

Порядок виконання практичної роботи

1. Використовуючи програмно-налагоджувальні засоби для МК51 або моделюючий комплекс *SCM* МК51 налагодити розроблену програму:

– сформувати початковий текст програми за допомогою текстового редактора, виконати пошук синтаксичних помилок та скомпілювати програму;

– налагодити розроблену програму.

2. Завантажити програму в УНС МК51. Переконатися в правильному виконанні індивідуального завдання.

3. Роздрукувати лістинг програми.

Зміст звіту

Звіт повинен містити короткі теоретичні відомості, схему алгоритму, налагоджену програму на асемблері, часові діаграми, а також висновки за роботою.

Контрольні питання

1. Пояснити структуру системи переривання МК К1816ВЕ51.
2. Пояснити призначення системи переривання, навести приклади застосування.
3. Пояснити призначення регістрів управління.
4. Навести принцип розподілу пам'яті в МК 1816ВЕ51.
5. Охарактеризувати особливості переривань від таймерів та послідовного приймача-передавача.
6. Як здійснюється апаратне усунення брязкоту контактів схем з TTL і КМОП?
7. Як здійснюється програмне усунення брязкоту контактів?
8. В чому необхідність застосування програмного або апаратного усунення брязкоту контактів?

8.2. Задачі для самостійного розв'язування

Задача 8.1. Привести приклад перемикання банків регістрів внутрішньої пам'яті даних. Розробити для МК51 програму обчислення різниці двох шістнадцятирозрядних чисел.

Задача 8.2. Розробити для МК51 програму для пересилки вмісту регістрів третього банку регістрів в резидентну пам'ять даних. Виконати сортування масиву даних у порядку збільшення.

Задача 8.3. Розробити для МК51 програму для перевірки на парність кількості одиниць у регістрі *R*3 другого банку регістрів. Встановити ознаку *F*0 якщо кількість одиниць непарна.

Задача 8.4. Привести приклад для МК51 застосування в програмах ознак *F*0 та *F*1. Розробити програму обчислення виразу, за виконання заданої умови.

Вихідні дані:

- Вираз для обчислення:

а). $F = R7 + FFh$, за умови, що $(R5 + R3) > 0$;

б). $F = R5 - R6 - FFh$, за умови, що $(R4 - R2) < 0$;

- Область виконання обчислення:

в). під час обчислення використовувати регістри першого банку регістрів загального призначення;

г). під час обчислення використовувати регістри другого банку регістрів загального призначення;

Задача 8.5. Розробити для МК51 алгоритм та програму для передачі управління на мітку *LLD,* якщо перемикач банку регістрів встановлений в наступний стан – біт *PSW*(4) = 1, біт *PSW*(3) = 0.

Задача 8.6. Розробити для МК51 алгоритм та програму обчислення заданого виразу, за умови встановлення ознаки *F*0/*F*1. Намалювати структурну схему підключення зовнішньої пам'яті даних.

Вихідні дані:

- Вираз для обчислення:

а). $R1 := 8(R5 - R4) + K$,

де $K = (R6 + 5)/4$, за умови $F0 = 1$,
$K = R1.R0 + R3.R2$, за умови $F0 = 0$;

б). $R7 := 4(R5.R4 - R0) + K$,

де $K = R2/4$, за умови $F1 = 1$,
$K = 3(R1 + R2)$, за умови $F1 = 0$;

- Область виконання обчислення:

в). під час обчислення використовувати регістри першого банку регістрів загального призначення;
г). під час обчислення використовувати регістри другого банку регістрів загального призначення;

- Місце розташування результату:

д). результат записати в комірки третьої сторінки зовнішньої пам'яті даних з адресами *C*0*h*. *A*4*h*;
е). результат записати в комірки другої сторінки зовнішньої пам'яті даних з адресами *C*6*h*. *A*8*h*;

Задача 8.7. Розробити для МК51 програму обчислення заданого виразу. Ємність сторінки ЗПД 2Кбайта. Намалювати структурну схему підключення до МК пам'яті даних.

Вихідні дані:

- Вираз для обчислення:

а). $F = 5(R5 - R3) + (R7 \,\&\, R6)/4$;

б). $F = (R2 - R7)/8 \vee 4(R3 + R6)$

- Область виконання обчислення:

в). під час обчислення використовувати регістри третього банку регістрів загального призначення;
г). під час обчислення використовувати регістри другого банку регістрів загального призначення;

- Джерело даних для обчислення:

д). дані для обчислення в регістри загального призначення заданого банку завантажуються з комірок третьої сторінки зовнішньої пам'яті даних з адресами *C*0*h*, *C*1*h*, *C*2*h*, *C*3*h*;

е). дані для обчислення в регістри загального призначення заданого банку завантажуються з комірок третьої сторінки зовнішньої пам'яті даних з адресами 23*h*, 24*h*, 25*h*, 26*h*;

- Ємність сторінки зовнішньої пам'яті даних:

ж). 2Кб;
з). 64Кб;

Задача 8.8. Розробити для МК51 програму реалізації алгоритму управління. Для вводу та виводу сигналів використати порт *P*1, *P*2 та *BUS*. Розряди портів *P*1[5], *P*1[4], *P*2[7], *P*2(6), *BUS*[1] та *BUS*[0] в початковому стані налагоджені на ввід інформації, а інші на вивід. Під час виконання алгоритму управління використати перший таймер/лічильник.

Вихідні дані:

- Алгоритми управління:

а). $\Pi\ (Y_4Y_5)X_1\overset{1}{\uparrow}Y_5\overset{1}{\downarrow}Y_2Y_3\ \text{K}$

Прийняти: $t_{Y_5} \geq 15 мкс$; $t_{Y_4} \geq 650 мкс$, $t_{Y_3} \geq 800 мкс$.

б). $\Pi\ (Y_1Y_2)X_1\overset{1}{\uparrow}(Y_3Y_4Y_5)X_2\overset{2}{\uparrow}Y_3\overset{2}{\downarrow}\overset{1}{\downarrow}\text{K}$

Прийняти: $t_{Y_1} = t_{Y_2} \geq 70 мкс$; $t_{Y_5} \geq 500 мкс$, $t_{Y_3} = t_{Y_4} \geq 420 мкс$.

в). $\Pi\ X_1\overset{1}{\uparrow}Y_2\overset{1}{\downarrow}Y_4Y_5X_2\overset{2}{\uparrow}Y_3\overset{2}{\downarrow}\text{K}$

Прийняти: $t_{Y_4} = t_{Y_2} > 46 мкс$; $t_{Y_5} > 660 мкс$, $t_{Y_3} = 35 мкс$.

Задача 8.9. Розробити структурну схему підключення до МК51 програмованого периферійного адаптера 580ВВ55. Адреси додаткових портів *PA, PB, PC* входять в загальний адресний простір третьої сторінки зовнішньої пам'яті даних. Детально розробити селектор адреси.

- Розробити програму

а) пересилки масиву даних з семи слів з порту *C* у внутрішню пам'ять МК, починаючи з комірки 50*h*;

в) з порту *A* переслати масив даних з десяти слів у внутрішню пам'ять МК, починаючи з комірки 3*Fh*;

с) в порт *B* переслати вміст регістрів другого банку регістрів;

д) в порт *C* переслати масив даних з внутрішньої пам'яті даних починаючи з комірки 44*h*.

Вихідні дані:

- Адреси портів:

е). *PA=F0h*, *PB=F1h, PC=F2h*, Регістр УСРР=*F3h*;

ж). *PA=C*8*h*, *PB=C*9*h, PC=CAh*, Регістр УСРР=*CBh*.

- Об'єм сторінок зовнішньої пам'яті даних – 64Кб.

Задача 8.10. Розробити для МК51 алгоритм обчислення заданого виразу. Для накопичування ознак переповнення, використовувати регістр *R*0.

Вихідні дані:

- Вираз для обчислення:

a). $F = 4(X1 - X3) \& (X2 + X4) - (X5 \vee X6)/2$;
b). $F = 16(X1 + X2 - 1) \& (X5 - X3)/4$;
c). $F = (X1 + 1) \& (X3 - X4)/4 - (X5 + X6)/2$;
d). $F = (X1 + X2) - 4(X3 \& X4) - 16(X5 \vee X6)$.

- Область виконання обчислення:

е). нульовий банк регістрів;
ж). перший банк регістрів;
з). другий банк регістрів;
к). третій банк регістрів.

- Місце розташування результату:

л). результат обчислення функції записати у регістри *R*7, *R*6 третього банку регістрів;
м). результат обчислення функції записати у регістри *R*6, *R*5 другого банку регістрів.

Задача 8.11. Розробити структурну схему підключення до МК51 зовнішньої пам'яті даних об'ємом 2Кб. Розробити програму пересилки масиву із двадцяти п'яти слів з другої сторінки пам'яті даних, розпочинаючи з комірки за адресою 80*h*. Перші

вісім слів переслати в третій банк регістрів, інші у резидентну пам'ять даних, розпочинаючи з комірки за адресою 40*h*.

Задача 8.12. Розробити для МК51 програму пересилки масиву із двадцяти слів із другої сторінки зовнішньої пам'яті даних, ємністю 2Кб, розпочинаючи з комірки за адресою *D*0*h*. Масив переслати у резидентну пам'ять даних, розпочинаючи з комірки за адресою 52*h*. Розробити структурну схему підключення до МК51 десяти сторінок зовнішньої пам'яті даних.

Задача 8.13. Розробити для МК51 програму пересилки масиву із тридцяти слів із резидентної пам'яті даних, розпочинаючи з комірки за адресою 50*h*, у третю сторінку зовнішньої пам'яті даних, розпочинаючи із адреси *A*2*h*. Об'єм сторінки зовнішньої пам'яті даних – 64Кб. Розробити структурну схему підключення до МК51 п'яти сторінок зовнішньої пам'яті даних.

Задача 8.14. Розробити для МК51 програму додавання трибайтних слів. Доданки розташовуються у комірках резидентної пам'яті даних за адресами 30*h*, 31*h*, 32*h*, розпочинаючи з молодшого байту. Початкові адреси доданків задавати в *R*0. Формат доданків в байтах задати в *R*3.

Вихідні дані:

- Місце розташування результату:

а). результат обчислення записати в першу сторінку зовнішньої пам'яті даних;
б). результат обчислення записати в другу сторінку зовнішньої пам'яті даних;

- Об'єм сторінки зовнішньої пам'яті даних:

в). 64Кб,
г) 2Кб,
д) 256б.

Задача 8.15. Розробити для МК51 програму обчислення різниці двох тридцятидвохрозрядних слів. Операнди розташовуються у комірках п'ятої сторінки зовнішньої пам'яті даних з наступними адресами: перший – *C*0*h*, *C*1*h*, *C*2*h*, *C*3*h*, другий – 03*h*, 04*h*, 05*h*, 06*h*, розпочинаючи з молодших байтів. Ємність однієї сторінки ЗПД 64 Кб.

Задача 8.16. Розробити для МК51 програму множення багатобайтних чисел. Вихідні дані розміщуються у зовнішній пам'яті даних.

Вихідні дані:

- Формат даних:

а). 16 байтів;
б). 24 байти;
в). 32 байти.

Задача 8.17. Розробити структурну схему МПС, у склад якої включити:

- п'ять сторінок пам'яті даних, першу сторінку виділити для зовнішніх пристроїв;
- пам'яті програм об'ємом 4Кб,
- додаткові порти – *P*5, *P*6, *P*7,
- програмований периферійний адаптер 580ВВ55 з адресами портів – *ACh*, *ADh*, *AEh*, *AFh*;
- децентралізовані КПП та КПДП;
- двадцять чотири зовнішніх пристрої, адреси портів яких вибрати самостійно.

Задача 8.18. Переслати дані із області даних зовнішньої пам'яті програм МК51 у регістри загального призначення.

Вихідні дані:

Поточна адреса пам'яті програм – 1044. Дані прочитати із комірок з адресами 2066 та 4077. Дані записати в регістри *R*5 та *R*7 першого банку регістрів.

Задача 8.19. Переслати дані із області даних зовнішньої пам'яті програм МК51 у регістри загального призначення.

Вихідні дані:

Поточна адреса пам'яті програм – 44. Дані прочитати із комірок з адресами 1066 та 3087. Дані записати в регістри *R*3 та *R*5 першого банку регістрів.

9. МОДУЛЬ 4

КУРСОВЕ ПРОЕКТУВАННЯ

Мета курсового проектування: Курсовий проект виконується за індивідуальним завданням і є самостійною роботою студента. Він призначений для розширення, закріплення, узагальнення і практичного застосування знань, умінь і навичок, отриманих студентом при вивченні курсу. В процесі курсового проектування студент повинен також навчитися користуватися довідковою літературою і вивчити процес створення проектно-конструкторської документації відповідно до діючих стандартів.

Завдання на курсове проектування

Розробити мікропроцесорну систему з урахуванням елементної бази, відповідно до варіанту. Виконати оцінку ефективності ухвалених технічних рішень.

До складу МПС повинні входити процесор (П), основна пам'ять (ОП), що містить ОЗП і ПЗП, зовнішні пристрої (ЗП), контролери переривань і контролери прямого доступу до пам'яті.

В курсовому проекті необхідно розробити три креслення – структурну схему мікропроцесорної системи, функціональну та принципову схеми пристроїв МПС відповідно варіанту.

Початкові дані для розробки МПС обрати з табл. 8.1 – табл. 8.14. У зазначених таблицях значеннями $h_8...h_1$ позначені молодші розряди номера залікової книжки, поданої в двійковій системі числення. Окрім того необхідно отримати у викладача індивідуальне завдання для кожного студента, складене за формою наведеною у додатку Є.

Таблиця 1.1. Варіанти завдань

h_3	h_2	h_1	**Вибір елементної бази**
0	0	0	1816 ВЕ48
0	0	1	1816 ВЕ51
0	1	0	1804 ВС1
0	1	1	*PIC* 16С84
1	0	0	1816 ВЕ51
1	0	1	1816 ВЕ48
1	1	0	*PIC* 16С54
1	1	1	*PIC* 16С74

Таблиця. 8.2. Варіанти завдань

Система команд (можливий прототип)
Комплексна (*Intel, VAX, Motorola*)

Проектування мікропроцесорної системи на основі секціонованих інтегральних схем

Структуру та ємність модулів ОП та кількість ЗП вибрати з табл. 8.3. Побудувати комутатор для вибору з пам'яті слів різної довжини.

Обчислити задану функцію для МК1804ВС1. Розробити цифрову діаграму виконання операції множення, операційну та функціональну схеми пристрою множення, алгоритм обчислення функції відповідно заданого варіанту (табл. 8.4), програму у символічних кодах мікроасемблеру.

Для реалізації системи переривань та режиму прямого доступу до пам'яті під час проектування МПС із табл. 8.5 обрати тип КПП і КПДП та структуру системної магістралі (СМ).

Завдання для розробки схем електричних функціональної та принципової обрати з табл. 8.6.

Таблиця. 8.3. Варіанти завдань

h_5	h_1	**Ємність пам'яті, ємність модуля пам'яті**	**Довжина слів для вибірки**	**Кількість ЗП**
0	0	2Мб, 512Кб	2*W*, *W*ст, *B*0, *B*2, *B*1	24
0	1	4Мб, 256Кб	2*W*, *W*мл, *B*3, *B*2, *B*1	56
1	0	8Мб, 512Кб	*W*ст, *W*ст, *B*0, *B*2, *B*1	72
1	1	4Мб, 128Кб	*W*мл, *B*3, *B*2, *B*1	68

Примітка: 2*W* – подвійне слово (32 розряди), *W*ст – слово (16 розрядів), *W*мл і *W*ст – молодша і старша частини подвійного слова (16 розрядів), *B*3 – *B*0 – байти з номерами 3 – 0 (8 розрядів).

Таблиця. 8.4. Варіанти завдань

h_7	h_3	**Спосіб множення**	**Функція для обчислення**
0	0	перший спосіб	$F = (R5 - R7) \times (R6 + 5)/4 - FFh$
0	1	другий спосіб	$F = (R2 \times R7)/8 \,\&\, (R6 + 5)/4$
1	0	третій спосіб	$F = 2 + (R1 \times R4)5 - (R6 + R2)/2$
1	1	четвертий спосіб	$F = (R0 \times R7)/4 + (R6 - 22h) \times 4$

Таблиця. 8.5. Варіанти завдань

h_1	h_2	**Тип КПП, КПДП**	**Організація СМ**
0	0	Централізований	ША, ШД
0	1	Централізований	ШАД
1	0	Децентралізований	ША, ШД
1	1	Децентралізований	ШАД

Примітка: ША, ШД – розділені шини адреси та даних, ШАД – суміщені шини адреси та даних.

Таблиця 8.6. Варіанти завдань

h_3	h_5	h_2	**Пристрій для розробки схем електричних функціональних та принципових**
0	0	0	БОД (розрядність даних – 32); схема прискореного переносу; шинні формувачі; схема управління станами та зсувами
0	0	1	КПП, КПДП, інтерфейс ЗП
0	1	0	БОД (розрядність даних – 32); схема прискореного переносу; шинні формувачі; схема управління станами та зсувами
0	1	1	БМУ (процесор 16-розрядний)
1	0	0	БМУ (процесор 24-розрядний)
1	0	1	БОД (розрядність даних – 32); схема прискореного переносу; схема управління станами та зсувами
1	1	0	КПП (кількість ЗП – 64)
1	1	1	Модуль ОП з вибіркою слів різної довжини; БМУ(автомат МУРА)

Проектування мікропроцесорної системи на основі однокристальних мікроконтролерів КР1816ВЕ48, КР1816ВЕ51, *PIC*

Для реалізації системи переривань та режиму прямого доступу до пам’яті під час проектування МПС обрати тип КПП та КПДП та структуру системної магістралі з табл. 8.7. Для МК1816ВЕ48 ємність пам’яті програм (ПП), пам’яті даних (ПД), модуля ОП та кількість ЗП вибрати з табл. 8.8.

Для МК 1816ВЕ51 об'єм ємність пам’яті програм, пам’яті даних та кількість ЗУ вибрати за табл. 8.9.

Для МК PIC16C84, PIC16C71 ємність пам'яті програм, пам'яті даних та кількості ЗУ обрати за табл. 8.10.

Адреси додаткових портів для мікроконтролерів 1816ВЕ48, 1816ВЕ51, PIC16 вибрати за табл. 8.11.

Під час розробки МПС розрядність операндів та спосіб виконання операції множення для всіх типів мікропроцесорів вибирати за табл. 8.12.

Таблиця 8.7. Варіанти завдань

h_1	h_2	**КПП, КПДП**	**Організація шини**
0	0	Централізований	ША, ШД
0	1	Централізований	ШАД
1	0	Децентралізований	ША, ШД
1	1	Децентралізований	ШАД

Таблиця 8.8. Варіанти завдань

h_3	h_2	h_1	**Ємність ПП**	**Ємність ПД**	**Кількість ЗП**
0	0	0	16 Кб	8 Кб	124
0	0	1	4 Кб	16 Кб	64
0	1	0	8 Кб	32 Кб	58
0	1	1	32 Кб	64 Кб	86
1	0	0	32 Кб	32 Кб	80
1	0	1	4 Кб	16 Кб	16
1	1	0	16 Кб	4 Кб	76
1	1	1	8 Кб	8 Кб	62

Таблиця 8.9. Варіанти завдань

h_2	h_3	h_1	**Ємність ПП (кількість сторінок пам'яті)**	**Ємність ПД (кількість сторінок пам'яті)**	**Кількість ЗП**
0	0	0	64 Кб (5 стор.)	6 Кб (10 стор.)	42
0	0	1	64 Кб (3 стор.)	64Кб (32 стор.)	80
0	1	0	64 Кб (2 стор.)	2 Кб (12 стор.)	127
0	1	1	64 Кб (6 стор.)	2 Кб (10 стор.)	64
1	0	0	64 Кб (3 стор.)	64 Кб (5 стор.)	76
1	0	1	64 Кб (2 стор.)	64 Кб (4 стор.)	48
1	1	0	64 Кб (6 стор.)	256 Б (16 стор.)	256

1	1	1	64 Кб (4 стор.)	2 Кб (20 стор.)	96

Таблиця 8.10. Варіанти завдань

h_3	h_2	**Ємність ПП, ПД (PIC16C84/54)**	**Кількість ЗП**	**Ємність ПП, ПД (PIC16C71)**	**Кількість ЗП**
0	0	2 Кб	42	4 Кб	53
0	1	4 Кб	60	2 Кб	39
1	0	8 Кб	54	8 Кб	65
1	1	1 Кб	38	2 Кб	24

Таблиця 8.11. Варіанти завдань

h_5	h_4	h_2	**Вибір додаткових портів**		
0	0	0	*P*4, *P*5	BB55	*A*0*h*, *A*1*h*, *A*2*h*, *A*3*h*
0	0	1	*P*4, *P*6, *P*7	BB51	20*h*, 21*h*, 22*h*, 23*h*
0	1	0	*P*7, *P*4	BB55	40*h*, 41*h*, 42*h*, 43*h*
0	1	1	*P*4, *P*5	BB55, BB51	50*h*, 51*h*, 52*h*, 53*h*
1	0	0	*P*5, *P*6, *P*7	BB55	0*Ah*, 1*Ah*, 2*Ah*, 3*Ah*
1	0	1	*P*4, *P*5, *P*6, *P*7	BB51	20*h*, 21*h*, 22*h*, 23*h*
1	1	0	*P*5, *P*7	BB51, BB55	70*h*, 71*h*, 72*h*, 73*h*
1	1	1	*P*7, *P*4	BB51	2*Ch*, 2*Dh*, 2*Eh*, 2*Fh*

Таблиця 8.12. Варіанти завдань

h_2	h_3	**Спосіб множення**	**1816ВЕ48, PIC16C84**	**1816ВЕ51**
			Розрядність операндів	**Розрядність операндів**
0	0	перший	24	24, з плаваючою комою
0	1	другий	16	32, з фіксованою комою
1	0	третій	8	24, з плаваючою комою
1	1	четвертий	16	24, з фіксованою комою

Функцію для обчислення сформувати залежно від коефіцієнтів приведених у табл. 8.13.

Функція для обчислення

$$X = K_1(A+B) + K_2(A*B) + K_3(A^2*B) + K_4(\frac{A}{B}) + K_5(A^2+B^2) +$$

$$+ K_6(\sqrt{A}*B) + K_7(B^2 + A*\sqrt{B}) + +K_8\frac{\sqrt{A}}{B-A} + K_9(A-B)$$

Таблиця 8.13. Варіанти завдань

$h_5\ h_4\ h_3\ h_2\ h_1$	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9
00001	1	0	0	1	0	0	0	0	1
00000	0	1	1	0	1	0	0	0	0
00011	0	1	0	1	0	0	0	0	1
00010	1	0	0	0	1	0	0	1	0
00101	0	0	1	0	0	0	1	0	1
00100	0	1	1	1	0	0	0	0	0
00111	1	1	1	0	0	0	0	0	1
00110	1	0	0	0	1	1	0	0	0
01001	0	1	0	0	0	0	1	1	0
01000	0	0	0	0	1	1	1	0	0
01011	1	1	0	0	0	0	1	0	0
001010	1	0	0	0	0	1	1	0	0
001101	0	0	0	0	0	0	1	1	1
001100	0	0	0	0	0	1	1	1	0
001111	1	0	0	0	1	1	1	0	0
001110	0	1	0	0	0	1	0	0	1
010001	0	0	0	1	1	1	0	0	0
010000	1	1	0	0	0	0	0	0	1
010011	1	1	0	0	0	0	0	1	0
010010	1	1	1	0	0	0	1	0	0
010101	1	1	0	0	0	1	0	0	0
010100	0	0	0	1	1	0	0	0	1
010111	0	0	0	1	0	0	1	0	0
010110	0	0	1	1	1	0	0	0	0
011001	1	1	0	0	0	1	1	0	0
011000	1	0	1	0	0	0	1	0	0
011100	0	1	0	0	0	0	1	1	0
011010	0	0	1	0	0	0	1	1	0
011011	1	0	1	0	1	0	0	0	0
011100	1	1	0	0	1	1	0	0	0
011101	0	0	0	1	0	1	1	0	0
011110	0	0	1	1	0	0	0	0	1
011111	1	0	0	1	1	0	0	1	0

100000	0	0	1	1	0	0	1	0	0
100101	0	0	0	0	1	1	0	1	0
101010	0	0	1	0	0	0	1	1	0
110110	1	0	1	0	1	1	0	0	0

Завдання для розробки схем електричних функціональної та принципової обрати з табл. 8.14.

Таблиця 8.14. Варіанти завдань

$h_4\ h_1\ h_2\ h_1$	**Блоки для розробки функціональної та принципової схем**
0000	Інтерфейс ЗП – вводу; ПП; 580ВВ55; 1816ВР43
0001	Інтерфейс ЗП – виводу; КПП; 1816ВР43; 580ВВ55
0010	580ВВ55; інтерфейс ЗП; КПП
0011	КПП (КПП –1804ВН1; 1804ВР3)
0100	КПДП
0101	580ВВ55; ПП; КПП
0110	КПП; ПД; 580ВВ55
0111	Інтерфейс ЗП – виводу; 580ВВ55; КПП
1001	Інтерфейс ЗП; КПП; КПДП
1010	Інтерфейс ЗП – виводу; 580ВВ55; 1816ВР43; КПП
1011	Інтерфейс ЗП – виводу; КПДП; КПП
1100	580ВВ55; КПП; додаткові порти (1816ВР43)
1101	КПП
1110	два інтерфейси ЗП – виводу і вводу; КПП; КПДП
1111	Інтерфейс ЗП – вводу; КПДП; ПП; ПД

Завдання необхідно узгодити з викладачем і будь які уточнення внести в лист індивідуального завдання за формою наведеною у додатку Є.

Зміст курсового проекту

Курсовий проект повинен містити наступні документи (в порядку їх комплектування):

1. Опис альбому.
2. Титульний лист.
3. Зміст технічного завдання.
4. Технічне завдання.
5. Сторінка з написом в середині листа

«Документи технічного проекту».

6. Відомість технічного проекту.
7. Мікропроцесорна система, схема електрична структурна.
8. Блок (за варіантом), схема електрична функціональна;
9. Блок (за варіантом), схема електрична принципова;
10. Зміст пояснювальної записки.
11. Пояснювальна записка і додатки до неї.
12. Сторінка з написом в середині листа

«Робоча документація».

Всі скомплектовані документи приводяться до формату А4 і скріпляються в одну папку або альбом.

Зразки оформлення титульної сторінки індивідуального завдання та інших документів приведені у додатку Є.

Технічне завдання розробляється студентом на підставі початкових даних, вказаних в таблицях варіантів відповідно до ГОСТ15.001-73.

Зміст технічного завдання

– *Призначення* об'єкту проектування, в якому розкриваються його області застосування (круг вирішуваних задач).

– *Склад* пристроїв, в якому приводиться перелік основних складових частин проектованого пристрою відповідно до ієрархічного принципу його побудови.

– *Технічні умови*, що розділяються на загальні й окремі. У загальних вимогах вказуються умови експлуатації, вимоги відмовостійкості і електричного режиму роботи, комплектуючі вироби, умови зберігання, безпеки, експлуатації, транспортування і т.п. В окремих – обумовлюються основні структурні і алгоритмічні особливості проектованого пристрою, як наприклад, система команд, адресність команд, система числення, довжина машинних слів, форма представлення чисел, тривалість операцій, ємність пам'яті і будь-які інші технічні вимоги.

– *Вимоги надійності*, в яких вказують характеристики відмовостійкості і ремонтопридатності пристрою.

– *Конструктивні вимоги*, що визначають типи і номенклатуру використовуваних елементів, кріпильних і комунікаційних виробів, ступінь уніфікації і стандартизації окремих конструкторських рішень, габаритно-вагові показники, особливості технологічного процесу виготовлення проектованого об'єкту.

– *Етапи* проектування і терміни їх виконання.

– *Перелік* текстової і графічної документації.

Технічне завдання повинно бути підписане виконавцем, керівником проекту і викладачем нормоконтролером.

Зміст пояснювальної записки

Вступ

1. Розробка архітектури і формування системи команд МПС.

1.1 Система команд (формати команд, приклади та мікроалгоритми виконання команд).

1.2 Функціональне призначення й принцип роботи основних блоків, які включені в склад МПС.

1.3 Реалізація обміну даними між пам'яттю та МК в програмному режимі. Схеми підключення пам'яті програм і пам'яті даних до МК. Привести приклади обміну даними, наприклад, передачі масиву даних із зовнішньої пам'яті програм в банк регістрів та інші.

1.4 Реалізація системи переривань. Розробка КПП. Навести схеми, які ілюструють роботу вибраного блоку за варіантом.

1.5 Реалізація обміну інформацією із зовнішніми пристроями в режимі прямого доступу до пам'яті. Розробка КПДП. Навести схеми, які ілюструють роботу вибраного блоку за варіантом.

2. Програмна частина

2.1 Мікроалгоритми та мікропрограми етапів виконання команд.

2.2 Мікроалгоритми та мікропрограми обчислення заданої функції, окремо показати мікропідпрограми множення двох чисел.

3. Розробка схем електричних функціональної і принципової.

Висновки

Перелік літератури

Додатки

У *вступі* подати мету і завдання курсової роботи, актуальність та науково-практичне значення обраної теми, на підставі яких документів здійснюється розробка мікропроцесорної системи. Вступ призначений для загального розкриття теми проекту, має обсяг 1–2 сторінки.

В розділі 1 навести обґрунтування вибору системи команд МПС, формати команд і даних, програмну модель машини.

Подати опис структурної схеми МПС. Вказати склад і призначення пристроїв і вузлів МПС, обґрунтувати вибір розрядності регістрів і шин, пояснити організацію модулів пам'яті тощо.

Розділ повинен містити опис роботи МПС в різних режимах, взаємодії пристроїв в процесі функціонування МПС.

У розділі також описати організацію апаратних, програмних і мікропрограмних засобів реалізації переривань, алгоритми взаємодії процесора і контролера переривань.

В розділі 2 навести функціональні мікроалгоритми виконання команд в термінах змістовних мікрооперацій – по одній команді з кожної з класу команд:

– системні команди;

– команди вводу-виводу;

– команди перетворення даних;

– команди передачі управління.

В розділ 3 навести опис розроблених функціональної і принципової схем.

У висновках узагальнити результати роботи.

У додатках до пояснювальної записки розмістити матеріали, що не входять у склад конструкторської документації (алгоритми, мікроалгоритми, програми, мікропрограми і таке інше).

Вимоги до оформлення курсових проектів

Порядок побудови розділів і підрозділів пояснювальної записки, правила викладення тексту, розрахунків, а також побудови таблиць повинні повністю відповідати вимогам єдиної системи конструкторської документації (ЕСКД).

Текст виконується в друкованому вигляді на аркушах формату А4 шрифтом *Times New Roman* 14 пунктів, міжрядковий інтервал 1,5 *Lines*. Всі заголовки, формули, цифри і таблиці повинні виконуватися шрифтом за ГОСТ 2.304-68.

Загальна специфікація повинна містити повний перелік конструкторських документів.

Перед комплектацією всі документи курсового проекту повинні бути підписані виконавцем і керівником проекту на титульному листі і в основних написах відомостей, специфікацій і креслень. Підпис керівника про допуск до захисту ставиться після остаточного оформлення альбому, що містить документацію за курсовим проектом.

Захист проводиться перед спеціальною комісією викладачів за безпосередній участі керівника проекту. Захист полягає в короткій доповіді (8 – 10 хвилин) студента і відповідях на питання. Студент повинен дати всі пояснення по суті проекту.

Якщо в конструкторській документації проекту буде виявлено грубі порушення ЕСКД або опиниться, що спроектований виріб принципово непрацездатний, проект оцінюється незадовільною оцінкою і повертається на доопрацювання.

ПЕРЕЛІК ЛІТЕРАТУРИ

1. Арифметичні та управляючи пристрої цифрових ЕОМ: Навчальний посібник / В.І.Жабін, І.А.Жуков, І.А.Клименко, Стиренко С.Г. – К.: ВЕК +, 2008. – 176 с.

2. *Бабич М.П., Жуков І.А.* Атестаційні роботи магістрів і спеціалістів*:* Навчально-методичний посібник. – К. НАУ, 2004. – 216 с.

3. *Бродин В.Б., Калинин А.В.* Системы на микроконтроллерах и БИС программируемой логики – М.: Издательство ЭКОМ, 2002. – 400 с.

4. *Дичка І.*А., Жабін В.І., Тарасенко В.П. Проектування обчислювальних систем х мікропрограмним управлінням. – К.: НТУУ «КПІ», 2001. – 53 с.

5. *Жабин В.И.* Архитектура вычислительных систем реального времени. – К.: ВЕК +, 2003. – 176 с.

6. *Жабин В.И., Ткаченко В.В.* Однокристальные и микропрограммируемые ЭВМ. – К.: Диалектика, 1995. – 116 с.

7. *Жабін В.І., Ткаченко В.В.* Цифрові автомати. Практикум. – К.: ВЕК +, 2004. – 160 с.

8. *Каган Б.М.* Электронные вычислительные машины и системы. – М.: Энергоатомиздат, 1985. – 552 с.

9. *Карцев М.А.* Архитектура цифровых вычислительных машин.– М.: "Наука", 1978. – 295 с.

10. *Мик Дж.* Брик Дж. Проектирование микропроцессорных устройств раз рядно-модульной организацией: В 2-х томах. Пер. С англ.. – М.: Мир, 1984. – 253 с.

11. *Микропроцессорные* системы: Учебное пособие для вузов / Е.К. Александров, Р.И. Грушевицкий, М.С. Куприянов и др.; Под общ. ред. Д.В. Пузанкова. – СПб.: Политехника, 2002. – 935 с.

12. *Молчанов А.А., Корнейчук В.И., Тарасенко В.П.* Справочник по микропроцессорным устройствам. – К.: Техніка, 1987. – 288 с.

13. *Новожилов О.П.* Основы микропроцессорной техники / Учебное пособие в двух томах. – М.: ИП РадиоСофт, 2007. – 336 с.

14. *Прикладана* теорія цифрових автоматів: Навчальний посібник / В.І.Жабін, І.А.Жуков, І.А.Клименко, В.В.Ткаченко. – К.: Книжкове видавництво НАУ, 2007. – 364 с.

15. *Проектирование* цифровых устройств на однокристальных микроконтроллерах / В.В. Сташин, А.В. Урусов, О.Ф. Мологонцева. – М.: Энергоатомиздат, 1990. – 224 с.

16. *Пухальский Г.И., Новосельцева Т.Я.* Цифровые устройства: Учебное пособие для втуов. – Спб.: Политехника, 1996. – 885 с.

17. *Пухальский Г.И.* Проектирование микропроцессорных систем: Учебное пособие для ВУЗов. – Спб.: Политехника, 2001. – 544 с.

18. *Разработка* вічислительных средств на базе микропроцессоров для систем числового программного управления: Отчет по НИР (науч. Рук В.И. Жабин), Киевский политехнический институт/№ ГР 01830078443, инв. № 02890031296. – Киев, 1989. – 253 с.

19. *Самофалов К.Г., Корнейчук В.И., Тарасенко В.П.* Цифровые ЭВМ. Теория и проектирование.– К.: Высш.шк. 1989. – 424 с.

20. *Сташин В.*В., Урусов А.В., Мологонцева О.Ф. Проектирование цифрових устройств на однокристальных микроконтроллерах. – М.: Энергоатомиздат, 1990. – 224 с.

21. *Тарабрин В.В., Лунин Л.Ф., Смирнов Ю.Н.* Интегральные микросхемы: Справочник. – М.: Радио и связь, 1990. – 528 с.

22. Учебно-отладочный стенд "EV8031/AVR". – OpenSystem – http://opensys.com.ua/Stend/Ev8031.

23. Учебно-отладочный стенд "EV8031/AVR". Методические указания к лабораторным работам. – OpenSystem, 2007. – 68 с.

24. Учебно-отладочный стенд "EV8031/AVR". Техническое описание. Инструкция по эксплуатации. – OpenSystem, 2007. – 26 с.

25. *Цифровые* ЭВМ. Практикум / К.Г.Самофалов, В.И. Корнейчук, В.П. Тарасенко, В.И.Жабин – К.: Высш.шк. 1989. – 124 с.

26. AT89C51. The technical report. – Atmel Corporation, 2000. – http://www.atmel.com/dyn/resources/prod_documents/doc0265.pdf/.

27. *ATmega*8515. The technical report. – Atmel Corporation, 2006. – http://www.atmel.com/dyn/resources/prod_documents/2512S.pdf.

ДОДАТКИ

ДОДАТОК А

ВКАЗІВКИ ДО ВИКОРИСТАННЯ МІКРОАСЕМБЛЕРУ

А.1. Трансляція мікропрограми

Мнемонічний двопрохідний мікроасемблер призначений для розробки мікропрограм.

Результатом роботи мікроасемблера є файл даних з розширенням «**.pmk*», який є вихідним для програмного емулятора системи на рис. А.1.

Процес трансляції вихідного файлу здійснюється за два проходи. Під час першого проходу відбувається визначення обсягу вихідного файлу, формуються таблиці міток і відповідностей, а також проводиться попередній синтаксичний аналіз. Під час другого проходу безпосередньо формуються коди мікрокоманд. У випадку виявлення синтаксичної або семантичної помилки в вихідному тексті процес трансляції припиняється з видачею повідомлення про характер помилки і рядка тексту, на якому вона була виявлена.

Вихідним файлом для мікроасемблеру є текстовий файл в кодах *ASCII* з розширенням «**.asm*». Розходження між заголовними і малими літерами мікроасемблером не сприймаються. Між окремими мнемоніками може бути будь-яке число службових символів, наприклад, пробіл, табуляція, повернення каретки, переклад рядка і таке інше.

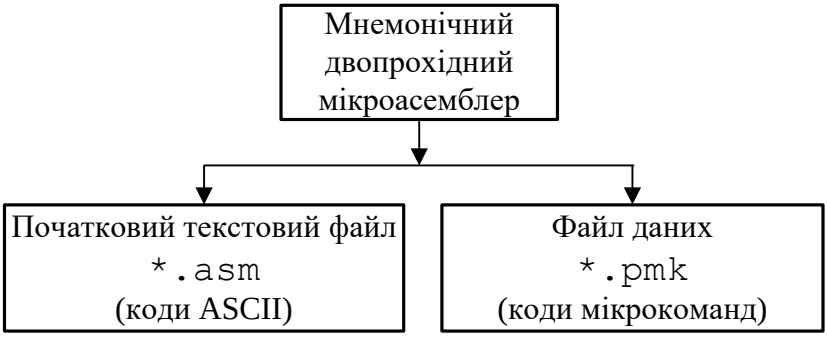


Рис. А.1. Процес трансляції мікропрограми

Строге дотримання правил написання мікропрограми, акуратність в наборі тексту прискорюють трансляцію і налагодження. Більшість помилок виникає насамперед через недбалий стиль написання і неточне знання самого об'єкта розробки.

Приклад текстового файлу (в кодах *ASCII*), який є початковим для мнемонічного двопрохідного мікроасемблера:

```
link l1:ct                          \ завдання зв’язків
accept r1:00ffh                     \ завантаження регістрів
org 100h                            \ розміщення МП у ПМК
macro inc reg:{add reg,reg,z,nz;}   \ створення макрокоманди
                                    \ інкременту
accept ra:12                        \ завантаження номеру
                                    \ регістру АЛП у RA
{add r10,ra;}                       \ R10:=R10 + R12
inc r1                              \ інкремент регістру R1
equ znach:125                       \ параметру znach
                                    \ присвоюється значення
                                    \ 125
{sub r1,znach,nz;}                  \ R1:=R1 – ZNACH
```

Коментарі

Коментарі використовуються для пояснень. Ознакою початку коментарю є символ «\ ». Далі мікроасемблер ігнорує всі символи, які зустрічаються, до наступного «\» або до кінця рядка.

Наприклад:

```
{add r11,r11,r10,z;} \додати до вмісту R11 вміст R10
```

Числові константи

Числові константи застосовуються під час завдання значень операндів і адрес. Ознакою константи є цифра на початку мнемоніки.

Наприклад:

```
\способи завдання констант у різних системах числення
65535              \десяткова константа
0FFFFh             \шістнадцятирічна константа
177777o            \вісімкова константа
1111111111111111%  \двійкова константа
```

Мітки

Мітки включають до 10 символів (букв, цифр та символів «_»), причому першим символом не повинна бути цифра. Ознакою кінця мітки служить будь-який зазначений далі розділювач: пробіл, повернення каретки, переведення рядка, табуляція. Мітка не повинна збігатися з зарезервованою мнемонікою. Приклади написання міток:

Наприклад:

```
\застосування міток
lll {or sll r1,r1,0;}   \логічний зсув вмісту регістру вліво
    {cjp not rm_z,lll} \якщо вміст регістру не дорівнює
                        \повторюємо зсув
    {add r1,r1,r3,z}    \додавання
    {cgp nz, lll}       \безумовний перехід на мітку
```

Мнемонічний запис мікрокоманд

Арифметичні мікрокоманди, що виконуються в АЛП, записуються в вигляді

```
<мнемоніка>(<оператор_зсуву>,)(<приймач_результату>,)
      <джерело_1>,<джерело_2>,<вхідний_перенос>
```

Тут і далі в круглих дужках вказуються необов'язкові елементи конструкцій.

Запис логічних мікрокоманд відрізняється від арифметичних відсутністю операнду вхідного переносу, тому що перенос не бере участь в логічних мікрооперація̈х.

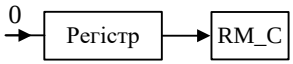
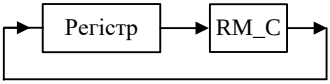
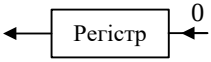
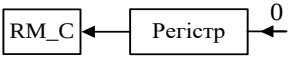
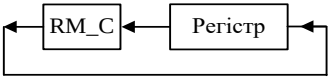
Мнемоніка арифметичних і логічних мікрокоманд зазначена в табл. А.1. Як приймач результату може бути зазначений кожний з регістрів *R*0 – *R*15, а також `nil`, коли результат в НОЗП не записується, але може бути виданий на локальну шину через БУ. Якщо приймач результату не вказується, то результат міститься на місце першого джерела операндів. Джерелами операндів можуть бути два регістри НОЗП, а також один регістр (він указується як перше джерело операндів) в комбінації з константою, `bus_d` або нулем. Нуль в полі джерела операнда позначається буквою `z`. Регістри НОЗП можуть адресуватися непрямо. Якщо в якості джерел операндів зазначені *RA* та/або *RB*, то операнди вибираються з регістрів, коди яких записані в *RA* і *RB*. Вхідний перенос може приймати значення 0, 1 (записується відповідно через `z` і `nz`), а

також `rm_c` і `not rm_c`. Мнемоніки операторів зсуву зазначені в табл. А.2.

Таблиця А.1. Мнемоніка мікрооперацій в АЛБ

Мнемоніка	Мікрооперація в АЛБ
`add`	$R + S + CI$
`sub`	$R - S - 1 + CI$
`or`	*R or S*
`and`	*R and S*
`nand`	*not*(*R and S*)
`xor`	*R xor S*
`nxor`	*not*(*R xor S*)

Таблиця А.2. Мнемоніка операторів зсуву

Мнемоніка	Найменування зсуву	Розряди МК $I_{10} - I_6$	Схема зсуву
`sra`	Зсув вправо арифметичний	00010	
`srl`	Зсув вправо логічний		
`sr.9`	Зсув вправо з переносом	01001	
`sla`	Зсув вліво арифметичний	10010	
`sll`	Зсув вліво логічний	10000	
`sl.25`	Зсув вліво з переносом	11001	

Приклади мікрокоманд:

```
{add slr,r10,r2,z;} \ R10 := 0.r[R1 + R10]
{xor r5,r5,r5;}     \ встановлення в нуль вмісту регістру R5
{SUB r7,r7,bus_d,nz;} \ віднімання з вмісту регістру R7
                      \ даних, що надходять з ЛШ
{add r9,z,nz;}        \ інкремент регістру R9
```

А.2. Директиви мікроасемблеру для блока обробки даних

Директиви мікроасемблеру – це службові мнемоніки, які не транслюються в мікрокоманди мікропрограми.

Вони служать для завдання початкових значень в регістрах, початкової адреси мікропрограми в пам‘яті мікрокоманд, для настроювання окремих вузлів обчислювальної системи тощо.

`accept` – ***директива занесення інформації в регістри БОД***

Загальний вигляд директиви:

```
accept <регістр>: <значення>
```

де `<регістр>`: *R*0, *R*1,…, *R*15, *RQ*,
POH,
RM, *RN*,
RA, *RB*.

Таким чином директива дозволяє задати `<значення>`:

- в будь-якому з регістрів АЛП,
- одночасно у всіх регістрах АЛП,
- значення ознак в регістрах *RM* та *RN* СУСЗ,
- значення в регістрах *RA*, *RB* обрамлення БОД.

Наприклад,

```
accept r1:0affh      \ R1:=AFFH
accept r1:0affh      \ R1:=AFFH
accept rq:12o        \ RQ:=12o
accept rm:1001%      \ RM:=1001%
accept rb:10         \ RB:=10
```

Директива

```
accept poh:<16 значень>
```

дає можливість задати значення в регістрах загального призначення АЛП. Після `poh` необхідно задати 16 значень, перше з яких завантажується в регістр *R*0, останнє – в регістр *R*15 АЛП.

Наприклад:

```
accept poh: 0,1,2,3,4,5,6,7,8,9,0ah,0bh,0ch,0dh,0eh,0fh
```

Наведемо приклади фрагментів мікропрограм із застосуванням директиви `accept`:

Приклад:

```
accept r2:123       \ завантажити значення 123 в R2
accept r10:375o     \ завантажити 8-кове значення 375 в R10
{sub r2,r10,nz;}    \ R2:=R2 – R10
```

Приклад:

```
accept rb:9         \ завантажити значення 9 в RB
{add rb, bus_d;}    \ виконати додавання вмісту регістру,
                    \ номер якого завантажений у RB із
                    \ значенням з локальної шини
                    \ R9:=R9+bus_d
```

`link` – *директива приєднання регістрів RA, RB до ЛШ*

Загальний вигляд директиви:

```
link <регістр>: <4 номери розрядів ЛШ>
```

де `<регістр>` – регістри *RA* або *RB* (рис. А.2).

Директива `link` вказує, номери розрядів 16-розрядної локальної шини які мають бути приєднані до виводів 4-розрядних регістрів *RA, RB* обрамлення БОД.

Наприклад, наступні директиви приєднають виводи *RA* до розрядів молодшої тетради локальної шини, а *RB* до розрядів третьою тетради ЛШ:

```
link rb:3,2,1,0
link ra:11,10,9,8
```

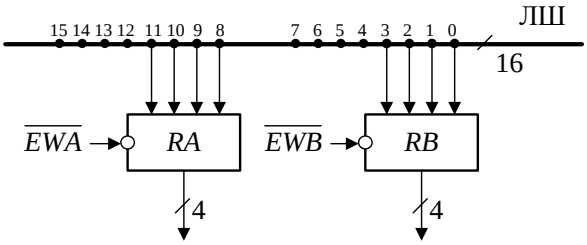


Рис. А.2. Приєднання регістрів *RA, RB* БОД до локальної шини

`load` – ***директива завантаження регістрів RA, RB з ЛШ***

Загальний вигляд директиви:

```
load <регістр>
```

де `<регістр>` регістри *RA* або *RB*.

Директива `load` означає, що в регістр (*RA* або *RB*) буде завантажено значення, яке в даний час перебуває на ЛШ.

Приклади застосування директиви `load`:

```
load ra
load rb
```

Приклад:

Фрагмент мікропрограми з використанням директив `accept`, `link`, `load`:

```
accept r3:0a2d4h      \ R3:=A2D4H
link rb:7,6,5,4       \ виводи регістру RB будуть
                      \ поєднані з розрядами другої
                      \ тетради ЛШ
{and nil,r3,00f0h;oey;} \ ЛШ:=00D0  A3C4
                      \ виконується мікрооперація
                      \ підсумовування вмісту
                      \ регістру R3 із константою,
                      \ результат не фіксується, але
                      \ видається на локальну шину (oey).
load rb               \ RB:=D  C
                      \ в регістр RB завантажується номер
                      \ регістра АЛП – R13
{sub rb,r10, z;}      \ R13:=R13 – R10 – 1
                      \ виконується мікрооперація
                      \ віднімання. Регістр RB адресує
                      \ перший з операндів. Місце
                      \ розташування результату задано
                      \ неявно. Результат записується за
                      \ місцем розташування першого
                      \ операнду.
```

Мікрокоманди управління регістрами RM* та *RN

Для завантаження регістрів *RM, RN* СУСЗ застосовуються наступні мікрокоманди

```
{load rm, z;}      \ встановлення всіх розрядів RM в нуль
{load rm, nz;}     \ встановлення всіх розрядів RM в одиницю
{load rm,flags;}   \ завантаження всіх ознак сформованих
                   \ під час виконання мікрооперації в АЛП
{load rn,flags;}   \ завантаження ознак у регістр RN
```

Одночасно з зазначеними мікрокомандами застосовуються мікрокоманди заборони запису у відповідні розряди *RM*, а саме `cem_c, cem_z, cem_n, cem_v.`

Приклад:

```
{load rm,flags;cem_v;cem_n;} \ завантаження тільки
                             \ розрядів rm_c та rm_z.
{load rm,flags;cem_c;}       \ завантаження всіх ознак
                             \ окрім rm_c.
```

`org` – *директива розміщення виконуваного коду мікропрограми в пам'яті мікрокоманд*

Загальний вигляд директиви:

```
org <мітка>
org <адреса>
```

Директива `org` розміщує виконуваний код мікропрограми в ПМК за вказаною адресою.

Приклади застосування директиви `org`:

```
org 20h
org start
```

Якщо мікропрограма не містить директиви `org`, вона розміщується в ПМК за адресою `000`.

`equ` – *директива завдання відповідності*

Загальний вигляд директиви:

```
equ <ім'я>:<значення>
```

Директива `equ` використовується для присвоєння символічним іменам, що використовуються у мікропрограмі, конкретних числових значень.

Приклади застосування директиви `equ`:

```
equ start:100
equ op1:2150
equ op2:0afh
```

`macro` – ***директива створення макрокоманд***

Загальний вигляд директиви:

```
macro<ім'я><формальні_параметри>:{<мікрокоманда>;}
```

Директива `macro` дозволяє конструювати власні мнемоніки операцій (макрокоманди) і користуватися ними надалі як стандартними.

Приклади застосування директиви `macro`:

```
macro inc reg:{add reg,reg,z,nz;}
macro dec reg:{sub reg,reg,z,z;}
macro mov reg1,reg2:{or reg1,reg2;}
```

Приклад:

```
inc r2          \ R2:=R2 +1
mov r10,r6      \ R10:=R6
dec rq          \ RQ:=RQ – 1
```

Ім'я макрокоманди надалі стає для транслятора звичайною стандартною мнемонікою. В якості імен формальних параметрів макроса не можуть бути застосовані зарезервовані мнемоніки. У мікропрограмі макрокоманда задається своїм власним ім'ям (мнемонікою) та реальними операндами, в тому ж самому порядку, в якому вони вказані в макросі.

`include` – ***директива приєднання файлу***

Загальний вигляд директиви:

```
include <ім'я_файлу>
```

Файл, що указується в директиві повинен знаходитись в одному і тому самому каталозі, що транслюється.

Приклад:

```
include macro.lib
include routine
```

А.3. Директиви мікроасемблера для блока мікропрограмного управління

`link` – *директива встановлення відповідності між входами МУ та логічними умовами*

Якщо в системі мікрокоманд схеми формування адреси мікрокоманди ФАМ, як умови використовуються сигнали на входах мультиплексора умов (МУ) – *L*1, *L*2, …, *L*6 (або *not L*1, *not L*2, …, *not L*6), то сигнал умови необхідно зв‘язати з одним із входів зазначеного мультиплексора (рис. 1) за допомогою директиви `link`. Під час аналізу *Li* (де $i = \overline{1,6}$) як логічної умови в структурі мікрокоманди у частині призначеній для управління ФАМ поле *MS* буде містити двійковий код відповідний до номеру входу умови – *i*.

Загальний вигляд директиви:

```
link <ім‘я_входу>:<умова>
```

де <ім‘я входу> відповідає *L*1, …, *L*6.

Приклади застосування директиви `link`:

```
link l2:rdm
link l3:no
link l4:ct
```

До входів *L*1, …, *L*6 МУ можна приєднувати наступні управляючі сигнали:

```
zo, co, no, vo
rm_z, rm_c, rm_n, rm_v
rn_z, rn_c, rn_n, rn_v
ct
rdm, rdd
int
irq0, ..., irq7
```

Завдання відповідності між входами МУ та умовами за допомогою директиви `link` може здійснюватись як окремо для кожного входу МУ, так і для всіх входів одночасно за допомогою наступної директиви

```
link l:<шість_умов>)
```

Приклад:

```
link l:ct,int,irq0,irq2,irq5,irq7
```

За цього до входів мультиплексора умов приєднаються відповідні управляючі сигнали *L*1:=*CT*, *L*2:=*INT*, *L*3:=*IRQ*0, *L*3:=*IRQ*2, *L*3:=*IRQ*5, *L*6:=*IRQ*7.

`accept` – *директива встановлення схеми ФАМ в початковий стан*

Директива `accept` дозволяє встановити в початковий стан наступні вузли ФАМ:

```
accept sp:<значення>          – покажчик стека
accept stack:< значення >     – комірки стека
accept rac:< значення >       – регістр адреси / лічильник
                                циклів (РА/ЛЦ)
accept pcmk:< значення >      – лічильник мікрокоманд
```

Приклад:

```
accept sp:003              \встановлення покажчика стека у
                           значення 3
accept stack[2]:0afdh      \записати в комірку 2 стека
                           \значення AFDH
accept stack:0a00h,0b00h,0c00h,2d0h,0d00h \5 значень
                           \заносяться в комірки стека
                           \комірка за коміркою
                           \розпочинаючи з верхівки
                           \стека
accept rac:5               \в РА/ЛЦ записати значення 5
accept rcmk:10             \початковий стан ЛМК – 10.
```

`link m` – *директива приєднання Буфера М до ЛШ*

Дванадцяти розрядний Буфер *M* поєднує 16-розрядну локальну шину та 12-розрядну шину адреси розгалуження (рис. А.3). Інформацію з ЛШ на ШАР можна подавати у довільному порядку послідовності бітів.

Директива `link m` задає відповідність між розрядами ЛШ та розрядами Буфера *M*.

Наприклад наступна директива

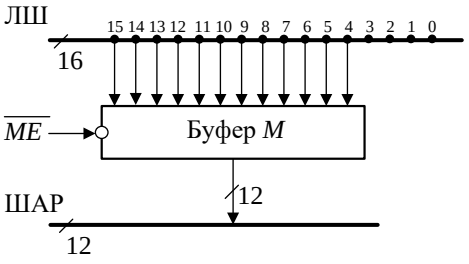
```
link m:15,14,13,12,11,10,9,8,7,6,5,4
```

з'єднає дванадцять старших розрядів ЛШ із входами Буфера М (рис. А.3, *а*).

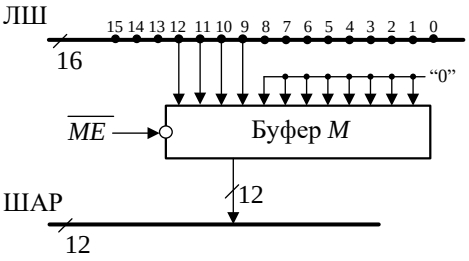
Директива

```
link m: 12,11,10,9,z,z,z,z,z,z,z,z
```

з'єднає 12, 11, 10, 9 розряди ЛШ із старшими чотирма входами Буфера М, інші розряди Буфера М завантажаться нулями (рис. А.3, *б*).



а



б

Рис. А.3. Поєднання локальної шини з Буфером *M*:
а, б – різні варіанти поєднання

`link v` – *директива налагоджування Буфера V*

Директива `link v` задає відповідність між управляючими сигналами, що поступають на вхід Буфера *V* та розрядами Буфера *V*.

На основі цих сигналів під час виконання мікрокоманди

```
{cjv cond;}
```

формується початкова адреса мікропрограми в ПМК, яка саме через Буфер *V* надходить в ФАМ.

На вхід Буфера *V* (рис. А.4) подаються наступні сигнали

`irq0, …, irq7` – запити на переривання від зовнішніх пристроїв,
`ct` – сигнал логічної умови,
`int` – сигнал вимоги загального переривання,
`rdm, rdd` – сигнали готовності пам‘яті та зовнішнього пристрою відповідно (у вигляді мнемонік ці сигнали позначаються без інверсії),
`z, nz` – сигнали “0” та “1”.

Наприклад наступна директива

```
link v:z,nz,irq0,irq1,irq2,irq3,irq4,irq5,irq6,irq7,ct,int
```

приєднує зазначені управляючі сигнали до дванадцяти входів Буфера *V* (рис А.4).

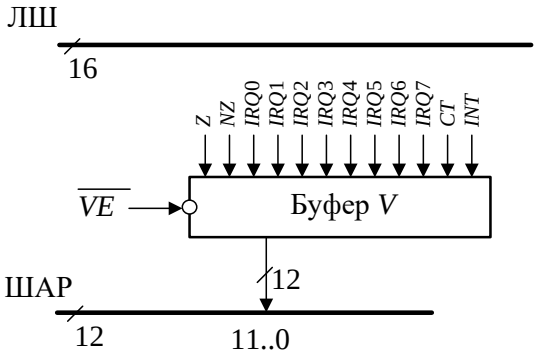


Рис А.4. Приєднання Буфера *V* до шини адреси розгалуження

А.4. Директиви роботи із пам’яттю та зовнішніми пристроями

dw - ***директива завдання значень комірок пам'яті має вигляд***

```
dw <адреса>:<значення>
```

Приклад:

```
dw 12:0ffh
dw 03Fh:15
```

Мікрокоманди управління зовнішніми пристроями

Мнемоніка мікрокоманд управління зовнішніми пристроями, пам'яттю, регістром адреси і буфером БУ збігається з найменуванням відповідного управляючого сигналу:

`r` – мікрокоманда читання з пам'яті;

`w` – мікрокоманда запису в пам'ять;

`i` – мікрокоманда вводу даних із зовнішнього пристрою;

`o` – мікрокоманда виводу в зовнішній пристрій;

`ewh`, `ewl` – мікрокоманди запису відповідно в старші і молодші розряди регістра адреси;

`oey` – мікрокоманда видачі результату *Y* з АЛБ на локальну шину.

Можна задати такі характеристики зовнішніх пристроїв (`dev`):

– тип пристрою (`in` - введення, `out` – виведення);

– адреса регістра стану (РС) в межах 64К (*max* 0*FFFFh*);

– адреса регістра даних (РД) в межах 64К (*max* 0*FFFFh*);

– затримка в тактах формування сигналу `rdd` (*max* 0*FFFFh*);

– затримка в тактах установки біта «Готовність» в регістрі стану (*max* 0*FFFFh*).

Приклад:

```
accept dev[2]:in,30h,32h,3,114
```

де `in` - пристрій вводу даних;

`30h` - адреса РС;

`32h` – адреса РД;

`3` – затримка сигналу *RDM* в тактах;

`114` – затримка установки біта готовності в РС після звертання до РД;

Для пристроїв введення можна задавати внутрішній буфер даних `dev_buf`, обсягом до 16 слів.

Приклад:

```
accept dev_buf[2]:1234h,5678h,89abh,0eeeeh
```

Зазначені після символу «:» дані вводяться в процесор один за одним під час кожного звертання до РД даного пристрою введення.

Директива `accept` дозволяє задати швидкодію пам'яті за допомогою змінної `rdm_delay`.

Приклад:

```
accept rdm_delay: 3
```

В даному випадку сигнал *RDM* буде формуватися з затримкою на 3 такти після видачі на шину керування сигналу *R* або *W*.

Для опису конфігурації зв'язків між компонентами системи використовується директива `link`.

Для установки відповідності входів *L*1, *L*2, *L*3 БМУ і логічних умов використовується директива

```
LINK <ім'я_входу>:<умова>
```

Приклад:

Для забезпечення зв'язків БМУ з пам’яттю, пристроями вводу/виводу та блоком обробки даних необхідно записати:

```
link L1:rdm
link l2:rdd
link l3:ct
```

Підключення двадцятирозрядного регістру адреси РАД до шістнадцятирозрядної ЛШ описується директивою:

```
link ewh : <номер_розряду>
```

Номер розряду РАД[19..0], зазначений у директиві, розділяє регістр на дві частини. Старша частина, включаючи зазначений розряд, управляється сигналом *EWH*, а молодша – *EWL*.

Приклад:

```
link ewh : 16
```

Директива набудовує зв'язки так, що за сигналом *EWH* чотири молодших розряди з ЛШ записуються в поле РАД[19..16]. За сигналом *EWL* всі шістнадцять розрядів з ЛШ записуються в РАД[15..0], тобто в молодшу частину регістра.

В один момент часу в різних вузлах системи можуть виконуватися різні мікрооперації. Всі мікрокоманди, що управляють мікрооперaціями, які виконуються в одному такті, записуються в операторних дужках { }, утворюючи повну мікрокоманду для даного такту роботи ЕОМ. Окремі мікрокоманди розділяються символом « ; », окрім того, під час запису мікрокоманди можуть використовуватися роздільники типу «пробіл», «повернення каретки», «переведення рядка». Повна мікрокоманда може займати кілька рядків в тексті мікропрограми. За необхідності мітка записується зліва перед операторною дужкою, що відкривається.

А.5. Приклади розробки мікропрограм

Приклад:

Встановити нулі в чотирьох старших розрядах РАД і записати в молодші розряди цього регістра адресу з *R*7. Прийняти слово з ОП в регістр *R*15. Сигнал готовності *RDM* формується рівнем логічного нуля.

```
\Область налагодження зв’язків
     link l1:rdm
     link ewh:16      \установка зв'язків між РАД і ЛШ
\Визначення затримки формування RDM
     accept rdm_delay:2
\-----------------------------------------------
\Область завантаження регістрів
\вихідна установка r7 (для налагодження)
     accept r7:1234h
\-----------------------------------------------
\Область завантаження пам’яті
\Завантаження даних в ОП за адресою 1234h
     dw 1234h:070fh
\-----------------------------------------------
\Область програми
     {cont;xor nil,r0,r0;oey;ewh;}    \РАД[19...16]:=0
     {cont;or nil,r7,z;oey;ewl;}      \РАД[15...0]:=r7
\Завантаження даних з ОП у регістр R15:=070fh
ll1  {cjp rdm,ll1;r;or r15,bus_d,z;}
     {}                           \кінець мікропрограми
```

Мікроінструкція БМУ `cont`, під час виконання якої відбувається формування адреси наступної мікрокоманди за інкрементом лічильника мікрокоманд (тобто відбувається лінійний перехід до наступної адреси), може бути безпосередньо не указана у мікрокоманді. При цьому формування наступної адреси відбувається за замовчуванням (розділ 2.5, табл. 2.27).

Приклад:

Підсумувати коди в регістрах *R*1,*R*2 і *R*15. Записати подвоєний результат в пам'ять за адресою, яка записана в регістрі, зазначеному в *RA*.

```
\Область налагодження зв’язків
     link l1:rdm
     link ewh:16
\Визначення затримки формування RDM
     accept rdm_delay:3
\-----------------------------------------------
\Область завантаження регістрів
     accept ra:3
     accept r3:0004h
     accept r1:4
     accept r2:16
     accept r15:32
\-----------------------------------------------
\Область програми
     {xor nil,r0,r0;oey;ewh;}        \РАД[19...16]:=0
     {add r1,r1,r2,z;}                    \ R1:=R1+R2
     {add sla,r1,r1,r15,z;}          \R1:=l(R1+R15).0
     {or nil,ra,z;oey;ewl;}       \запис адреси в РАД
\запис результату в пам'ять
ll2  {cjp rdm,ll2;w;or nil,r1,z;oey;}
     {}                          \кінець мікропрограми
```

ДОДАТОК Б

МОДЕЛЮЮЧИЙ КОМПЛЕКС *SCM* МК48

SCM (Single-Chip Machine) представляє собою систему моделювання роботи внутрішніх компонентів БІС сімейства МК58.

SCM МК48 призначена для:

– моделювання роботи мікроконтролера КР1816ВЕ48 в сукупності з мікросхемою-розширювачем портів вводу виводу КР580ВР43 і блоком зовнішньої пам'яті об'ємом 256 байт;

– розробки і налагодження програм для мікроконтролерів серії МК48;

– дослідження поведінки внутрішніх і зовнішніх сигналів вказаних мікросхем.

Головне вікно програми SCM зображено на рис. Б.1. У верхній частині вікна розміщене головне меню програми та інструментальна панель. Опис основних пунктів головного меню та інструментальної панелі дивись у додатку В.

У центральній частині головного вікна розміщуються функціональні пристрої, з яких складається МК48: два послідовні порти *P*1 та *P*2, паралельний порт *BUS*, блок управління БУ, таймер та АЛП. Структурі елементи поєднані між собою каналами зв'язку. У структурі кожного елементу МК48 зображені регістри та їх вміст. Під час виконання програми у моделюючому комплексі відображаються зміни вмісту регістрів відповідно до кожного шагу моделювання.

Для *завдання вихідних значень у буферах портів* необхідно натиснути піктограми (1) (рис. Б.1), розміщені біля кожного з портів. Під час цього відкривається діалогове вікно редагування буферу відповідного порту (рис. Б.2).

Наприклад, для встановлення вихідних значень у буфері порту *P*1, обираємо розділ діалогового вікна «**Буфер для P1**». Натисненням кнопки «**Добавить**», додамо один буфер і встановимо його значення в *C*0. Встановлення значення виконується безпосередньою зміною значень певних бітів. Щоб змінити значення біта на протилежний, необхідно виконати подвійний клік мишкою на певному біті.

Зліва у головному вікні програми відображена структура та вміст пам'яті програм МК48, справа – структура та вміст пам'яті даних

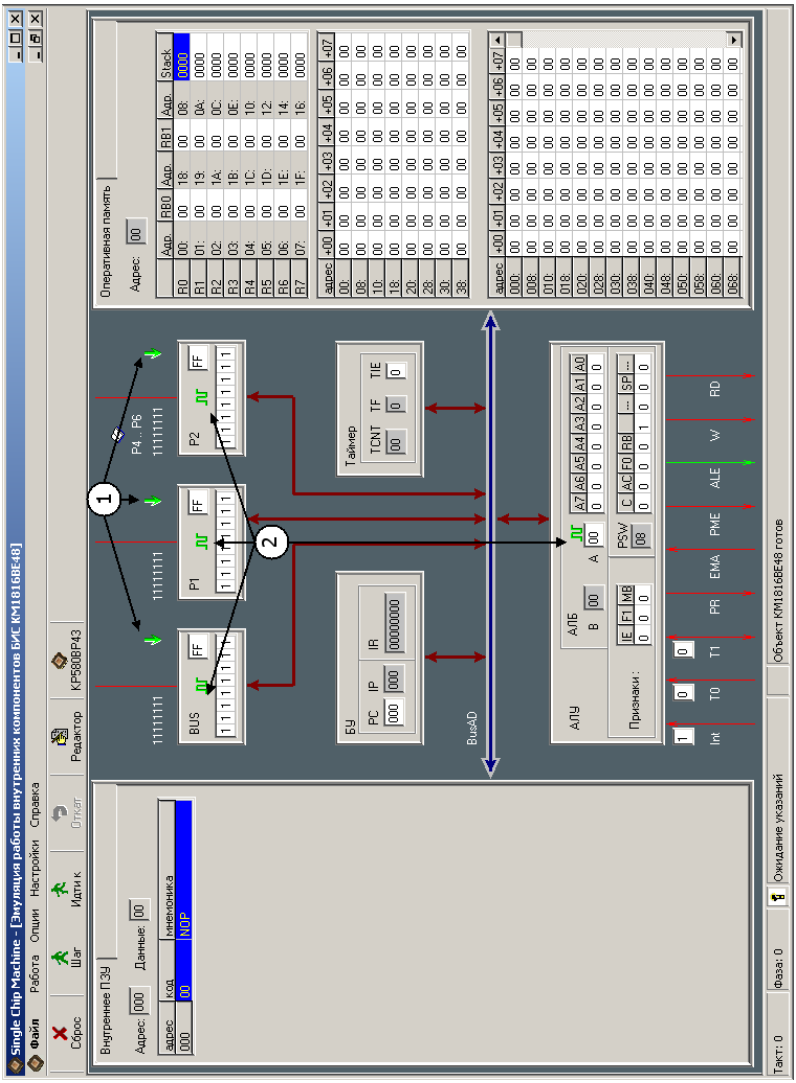


Рис. Б.1. Головне вікно програми *SCM1*

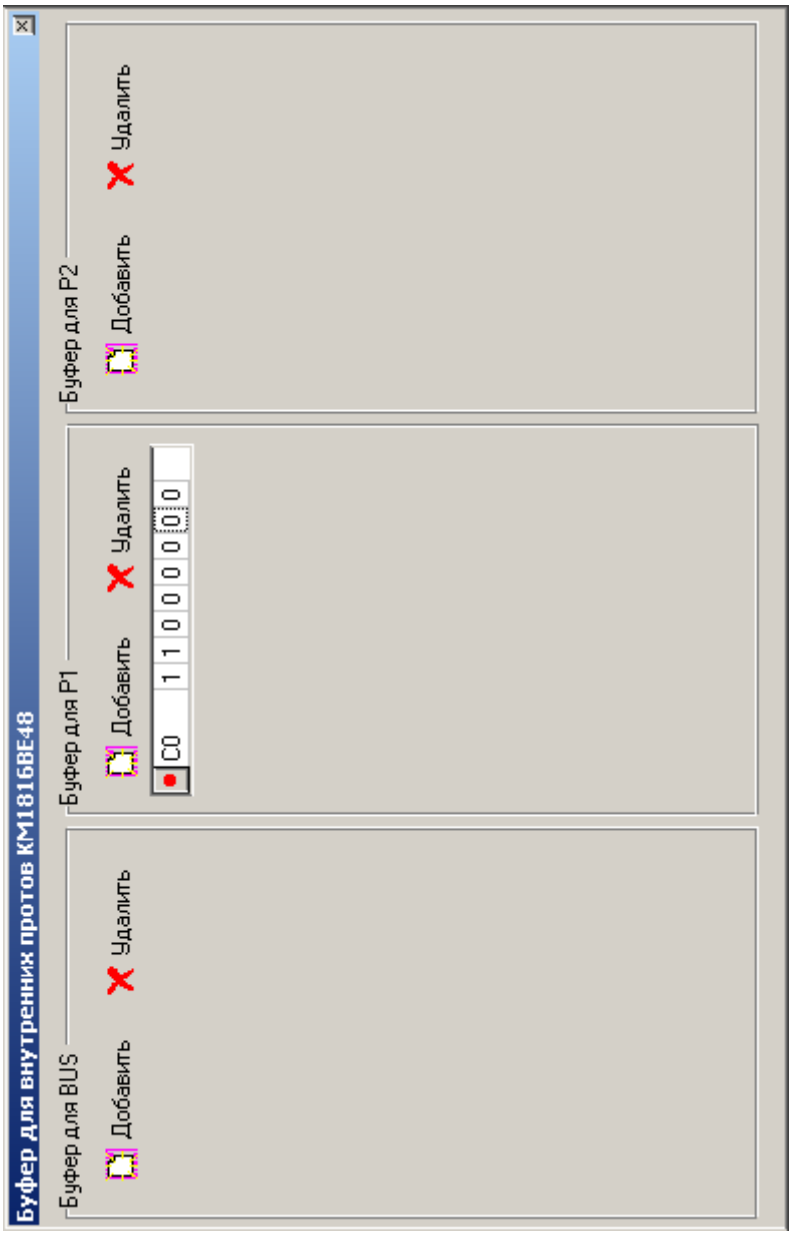


Рис. Б.2. Вікно редагування внутрішніх портів

Для *завантаження вихідного файлу* програми, використовують команду головного меню **Файл → Открыть**, під час цьому відкривається вікно редагування коду програми (рис. Б.3).

Для *збереження результатів редагування* програми необхідно застосувати команду головного меню **Файл → Сохранить**. Для компілювання програмами – команду головного меню **Компиляция → Компиляция** (клавіша <CTRL / F9>), для запуску **Компиляция → Запуск** (клавіша <F9>).

У випадку помилки під час компіляції вказується помилка і номер рядку, в якому вона відбулася. У випадку успішної компіляції, вікно редагування закривається, а в область пам'яті програм головного вікні завантажується вихідна програма. На початку виконання програми встановлюють значення вхідних буферів у вікні редагування портів(рис. Б.2).

Для покрокового *налагодження програми* використовується піктограма інструментальної панелі «**Шаг**». Головне вікно програми під час налагодження програми зображено на рис. Б.4.

Для виконання програми до першої поміченої команди (або кінця програми), використовується піктограма «**Идти к**». Позначка команд виконується кликом лівої кнопки миші на необхідній команді. При цьому обирається необхідна в закладка вікна пам'яті програм – «**Внутреннее ПЗУ**» або «**Внешнее ПЗУ**».

Зміни, що відбуваються з вмістом регістрів структурних елементів МК48 або комірок пам'яті даних під час покрокового виконання програми помічаються жовтим фоновим кольором (рис. Б.4).

Під час виконання програми можна продивитись *часові діаграми сигналів* пристроїв МК48. За натискання піктограм (2) (рис. Б.1) відкривається вікно часових діаграми відповідного пристрою. Часова діаграма сигналів порту *P*1 зображена на рис. Б.5. На діаграмі відображаються значення всіх бітів порту в кожний момент часу роботи МК48. Діаграма дозволяє скинути значення бітів, відключивши/включивши живлення, а також виконати побітовий пошук конкретного значення порту на діаграмі.

Вікно часових діаграм сигналів можна також відкрити за допомогою команди головного меню **Работа → Открыть диаграмму → Порта P1** (рис. Б.5).

Для визначення часу сигналу, необхідно виконати клік лівою кнопкою мишки на початку сигналу і правою в кінці. На рис. Б.5 визначена тривалість сигналу – 35 мкс.

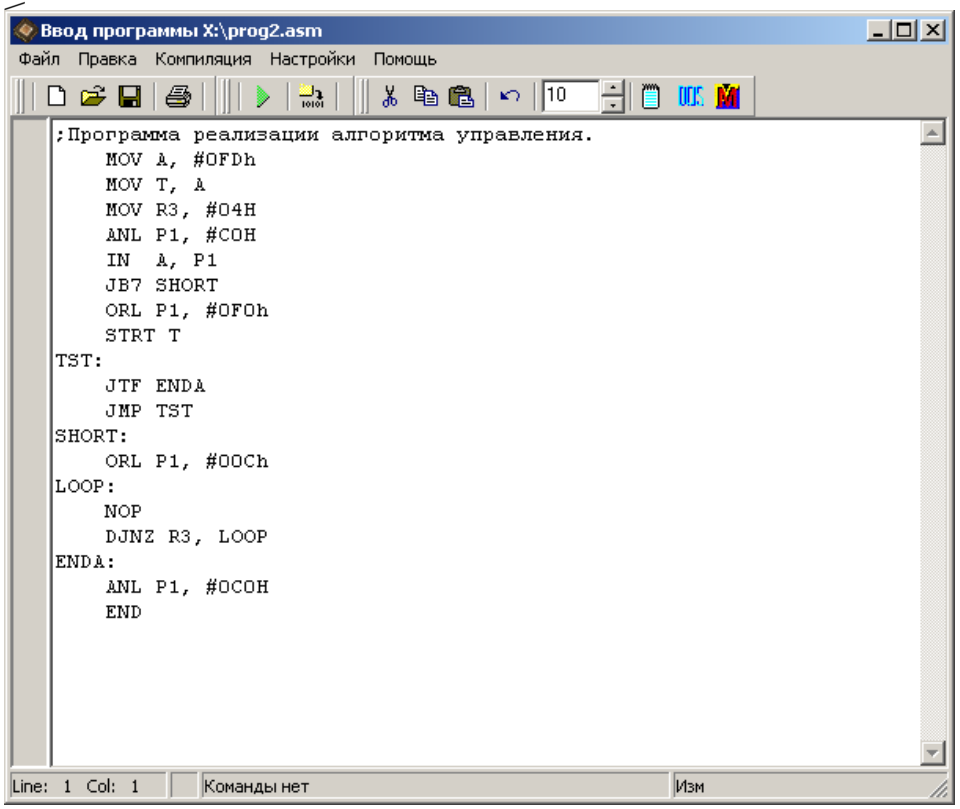


Рис. Б.3. Вікно редагування програми

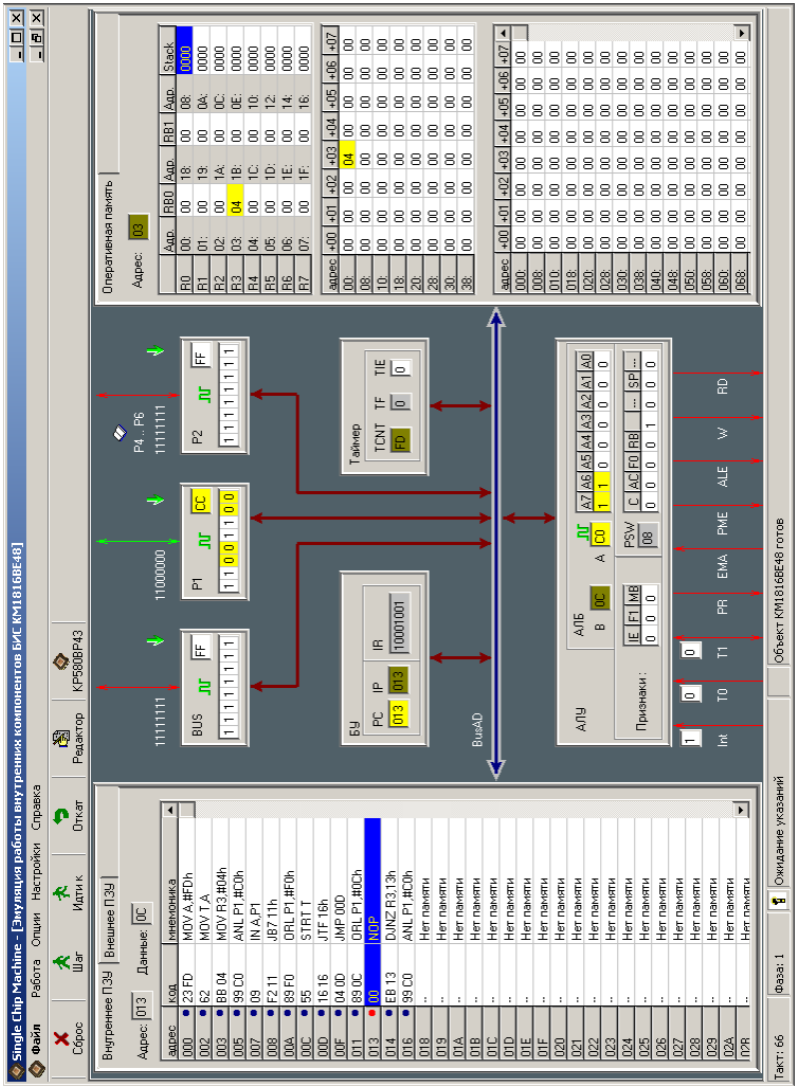


Рис. Б.4. Вікно налагодження програми

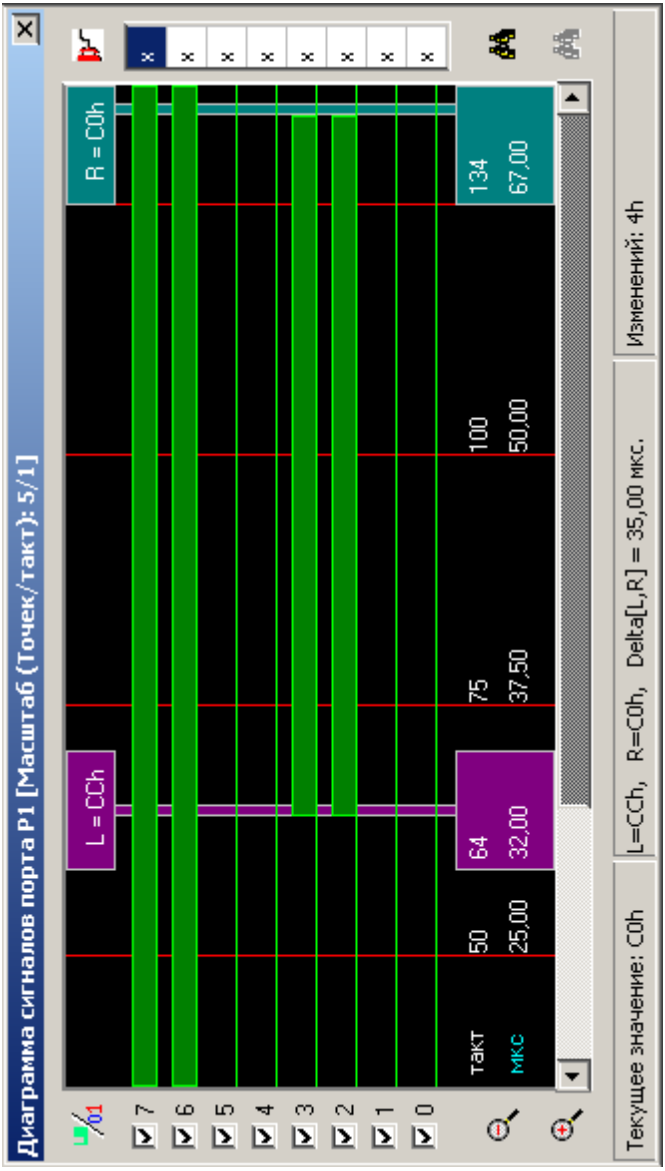


Рис. Б.5. Вікно перегляду часових діаграм сигналів

ДОДАТОК В

МОДЕЛЮЮЧИЙ КОМПЛЕКС *SCM* МК51

SCM (Single-Chip Machine) представляє собою систему моделювання роботи внутрішніх компонентів БІС сімейства МК51.

SCM МК51 призначена для:

– моделювання роботи мікроконтролера КМ1816ВЕ51 в сукупності з мікросхемою додаткових портів КР580ВР43 та блоком зовнішньої пам'яті даних об'ємом 256 байт;

– розробки на налагодження програм для МК51;

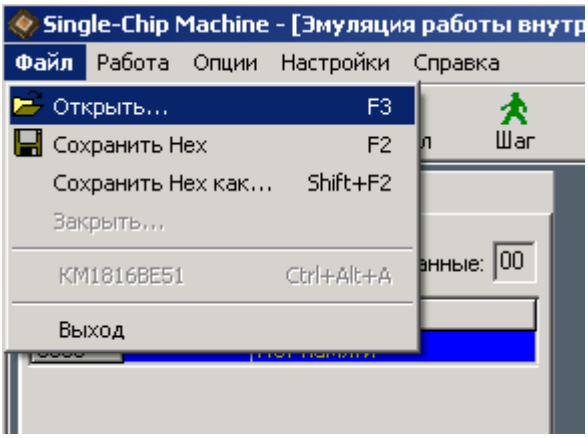
– дослідити поведінку внутрішніх і зовнішніх сигналів вказаних мікросхем.

Зовнішній вигляд головного вікна програми показаний на рис. В.1. У верхній частині вікна розміщене головне меню програми та інструментальна панель. Головне меню містить наступні команди:

– **Файл**
– **Работа**
– **Опции**
– **Настройки**
– **Справка**

Далі детально розглянуті команди головного меню.

Меню «Файл»



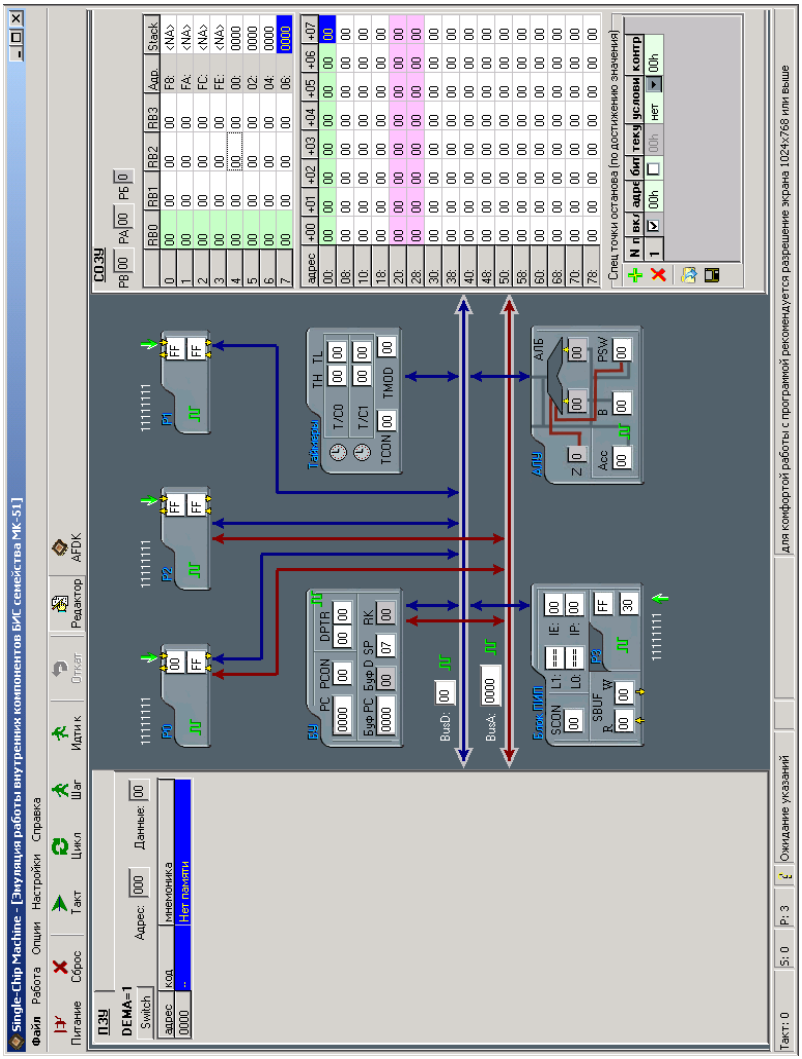


Рис. В.1. Головне вікно програми *SCM2*

– **Открыть** – викликає діалогове вікно відкриття файлу;

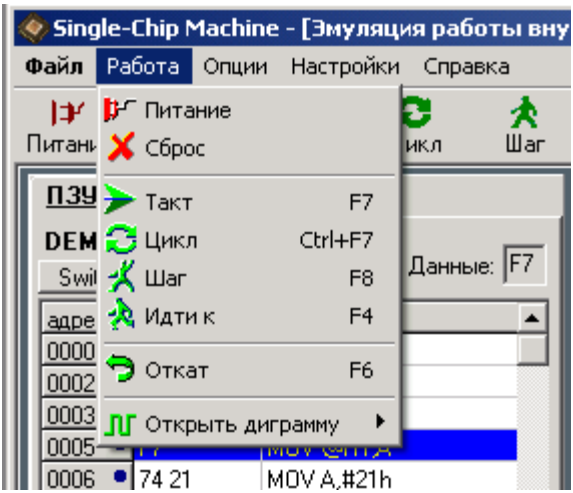
– **Сохранить *Hex*** – зберігає скомпільовану програму у стандартному форматі машинного подання пам'яті програм hex (так званий hex-файл);

– **Сохранить *Hex* как** – під час зберігання *hex*-файлу вказати шлях та нове ім'я файлу;

– **Закрыть** – закриває поточний файл;

– **Выход** – закриває програму та всі вікна.

***Меню* «Работа»**



– **Питание** – ініціалізація мікроконтролера;

– **Сброс** – припинення виконання програми та перехід на адресу 0000 пам'яті програм;

– **Такт** – відпрацьовування одного такту модельного часу;

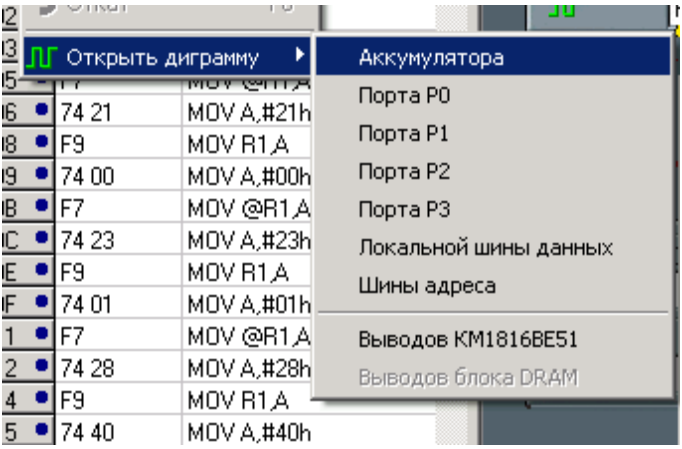
– **Цикл** – відпрацьовування одного циклу;

– **Шаг** – виконання поточної команди та перехід до обробки наступної команди;

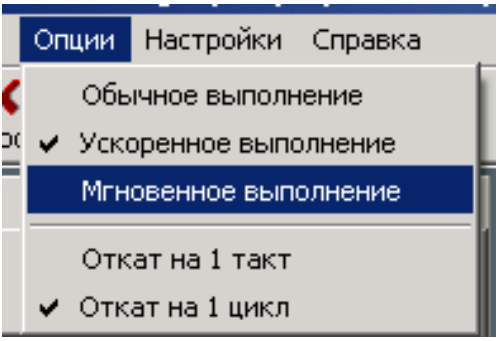
– **Идти к** – виконання всіх команд пам'яті мікрокоманд;

– **Откат** – повернення системи до стану, в якому вона була до попереднього такту/циклу (в залежності від налаштувань);

– **Открыть диаграмму** – в наступному контекстному меню можна вибрати необхідну для перегляду діаграму:



Меню «Опции»



– **Обычное выполнение,**

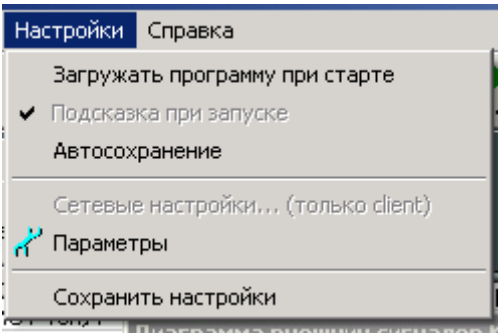
ускоренное выполнение,

мгновенное выполнение – швидкість роботи моделюючої програми;

– **Откат на 1 такт** – налаштувати програму так, щоб вона при виконанні відкату переходила у стан, в якому була 1 такт назад;

– **Откат на 1 цикл** – налаштувати програму так, щоб вона при виконанні відкату переходила у стан, в якому була 1 цикл назад.

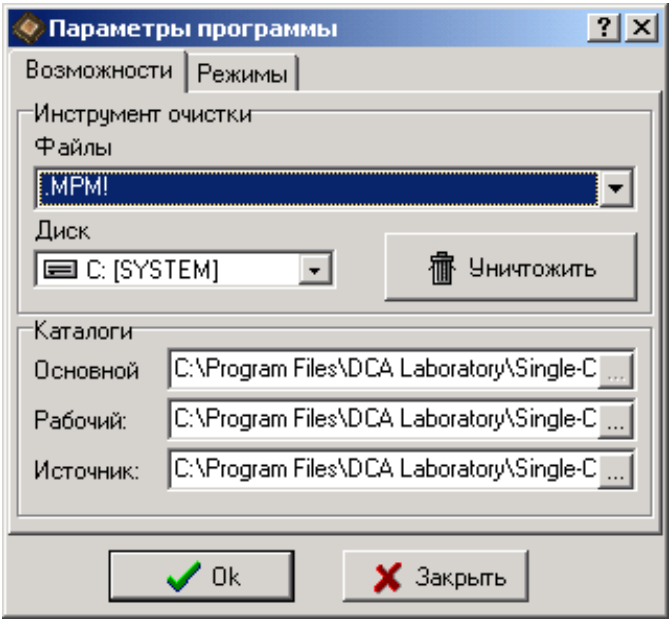
Меню «Настройка»



– **Загружать программу при старте** – автоматичне завантаження програми при включенні комплексу;

– **Автосохранение** – автоматичне збереження тесту програми під час його редагування;

– **Параметры** – відкриття діалогового вікна для налаштування параметрів програми:



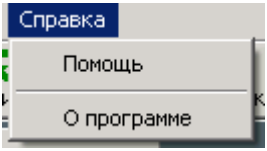
Инструмент очистки – використовується, щоб вказати розширення тимчасових файлів, для моделюючої програми *SCM* МК51, а також для їх очистки.

Каталоги – використовується для вказування шляхів до основних каталогів програми, якими є:

– **Основной** – шлях до інстальованої програми;
– **Рабочий** – каталог *Work* в основному каталозі;
– **Источник**– каталог *Source* в основному каталозі.

Увага!!! Після встановлення моделюючої програми на комп'ютер обов'язково правильно налаштувати значення основних каталогів, після чого зберегти параметри налаштування програми за допомогою кнопки **OK**.

***Меню «*Справка»**



– **Помощь** – викликає файл довідки

– **О программе** – інформація про програму *SCM* МК51 та її розробників.

Деякі з пунктів меню винесені на панель інструментів:

– ввімкнути/вимкнути живлення;

– припинення виконання програми та перехід на адресу 0000 пам'яті програм;

– відпрацьовування 1 такту модельного часу;

– відпрацьовування 1 циклу;

– виконання поточної команди та перехід до обробки наступної команди (виконання одного кроку);

– виконати всі команди пам'яті мікрокоманд, або послідовність команд до поміченої команди;

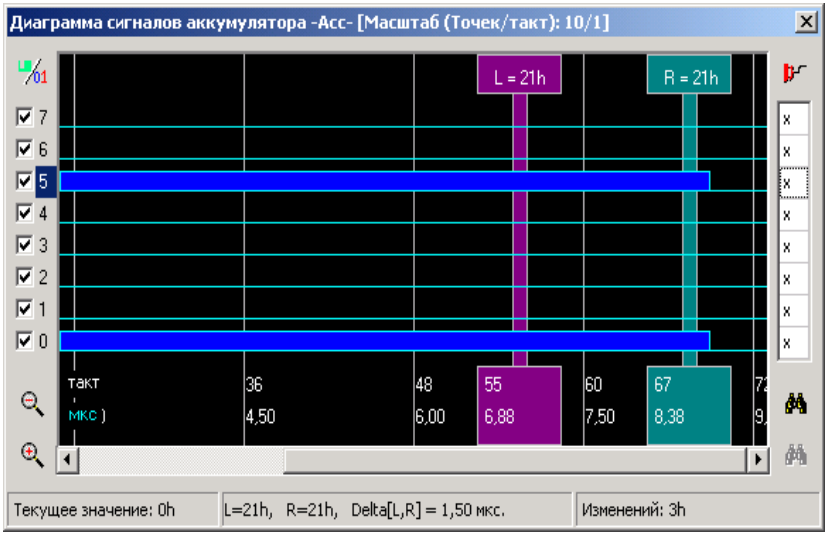
– повернення системи до стану, в якому вона була до попереднього такту/циклу (в залежності від налаштувань);

– редагування тексту програми для виконання на мікроконтролері;

– УГП ВІС сімейства МК51.

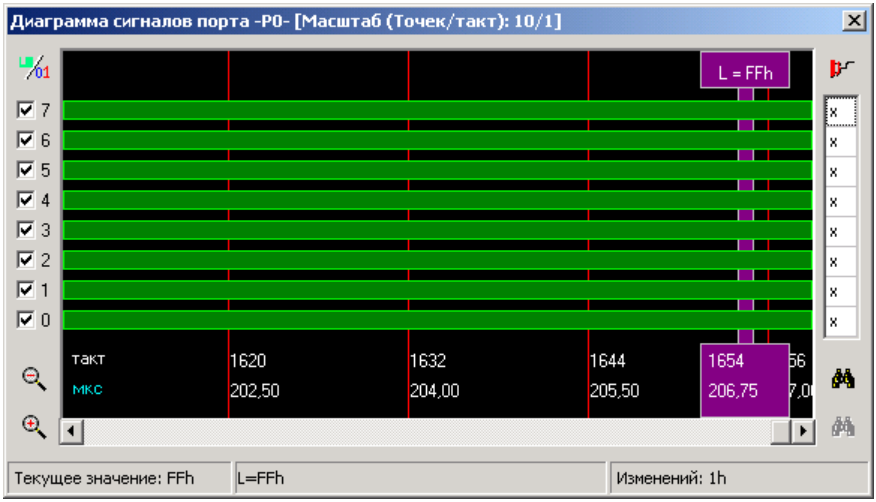
Робота з діаграмами сигналів

На діаграмі сигналів відображуються значення всіх бітів певних пристроїв (акумулятору, портів, локальних шин адреси та даних) в конкретній момент часу роботи МК51. Діаграма дозволяє скинути значення пристрою відключивши/включивши живлення, а також виконати побітовий пошук конкретного значення сигналу на діаграмі. На рис. В.2, зображені діаграми сигналів різних пристроїв МК51.

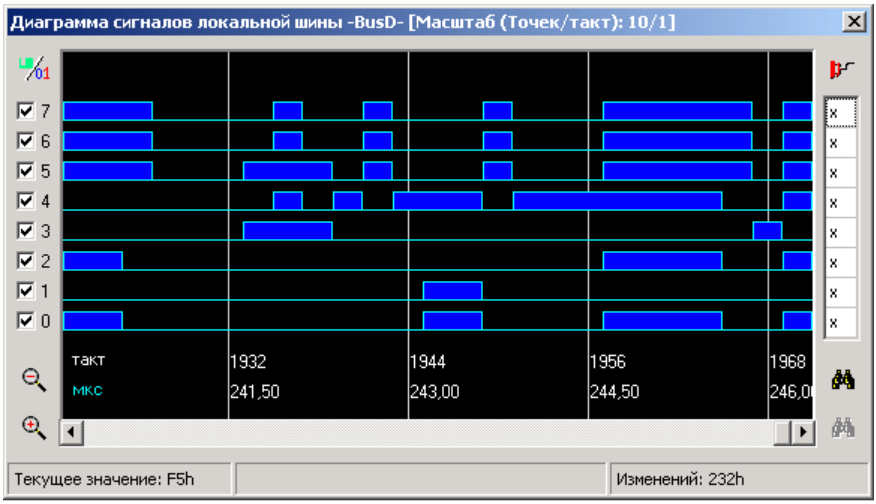


а

Рис. В.2. Діаграми сигналів: *а* – діаграма сигналів акумулятора; *б* – діаграма сигналів порту *P*0; *в* – діаграма сигналів локальної шини даних *BusD*; *г* – діаграма зовнішніх сигналів.

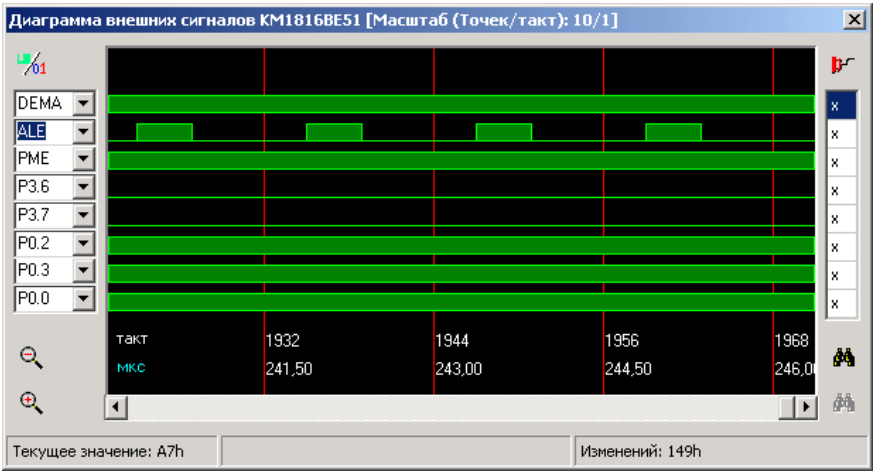


б



в

Продовження рис. В.2.



г

Закінчення рис. В.2.

В панелі статусу вікна діаграми сигналів відображуються наступні значення:

– поточне значення пристрою;

– інтервал часу між двома точками на діаграмі;

– кількість змін стану пристрою на діаграмі на поточний момент часу.

Для визначення інтервалу часу необхідно виконати наступні кроки:

1. Правою кнопкою миші вказати ліву границю шуканого інтервалу на діаграмі.

2. Лівою кнопкою миші вказати праву границю шуканого інтервалу на діаграмі.

3. В середньому відділі панелі статусу відображується значення модельного часу в першій точці (**`L =`**), значення модельного часу в другій точці (**`R =`**), та інтервал часу між ними (**`Delta [L, R] =`**) (рис. В.2, *а*).

Редагування тексту програми та запуск її на виконання

Під час натискання піктограми **`Редактор`** панелі інструментів відкривається вікно редактора-компілятора (рис. В.3).

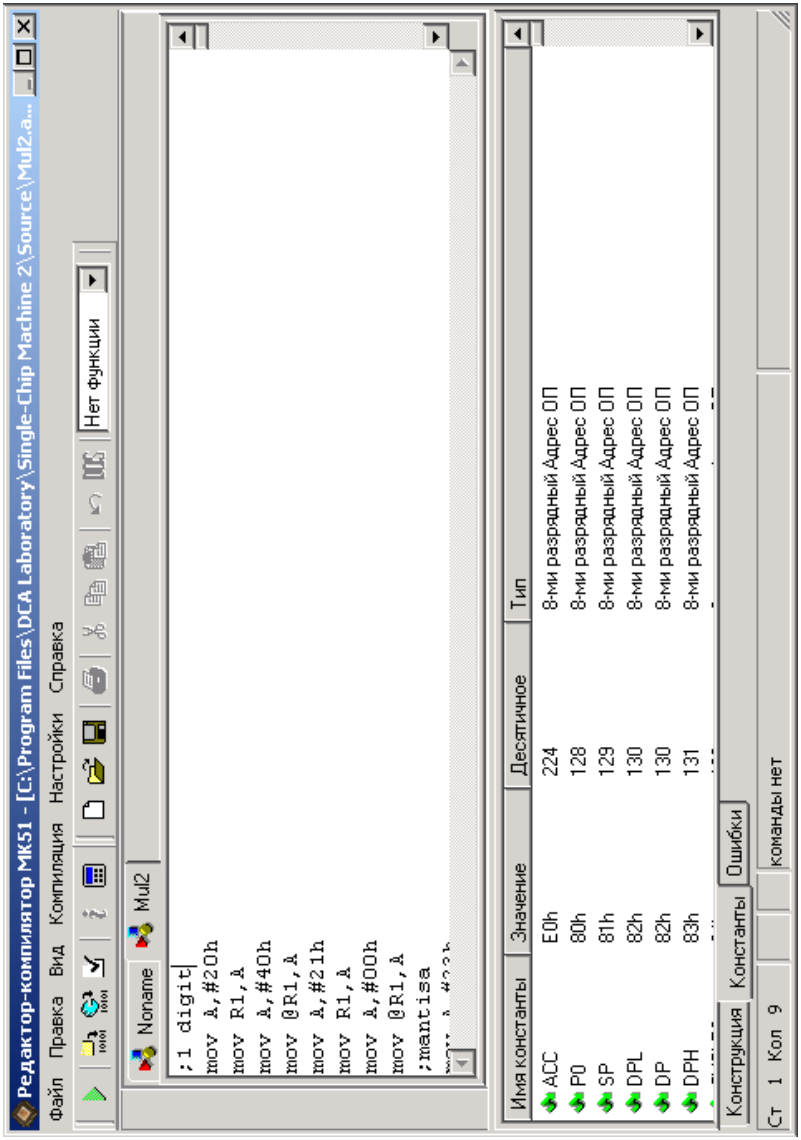
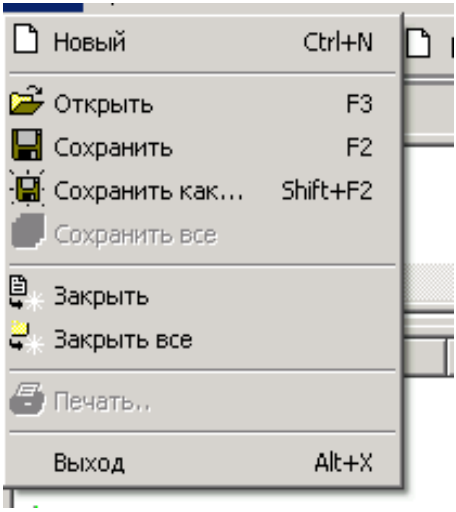


Рис. В.3. Вікно редактора-компілятора

За допомогою команди головного меню «**Файл**» можна або створити новий файл з текстом програми – команда

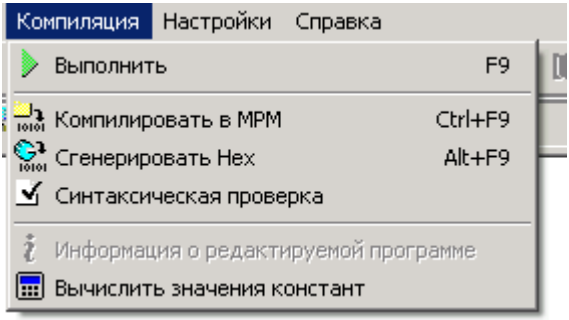
Файл → Новый, або відкрити вже існуючий – **Файл → Открыть** (рис. В.4).



Зберегти відредаговану програму можна за допомогою команд **Файл → Сохранить** або **Файл → Сохранить как.**

Команди **Файл → Закрыть, Файл → Закрыть все,** застосовуються для закриття робочого файлу або всіх відкритих файлів відповідно. Команда **Файл → Выход** закриває вікно редактора компілятора.

Відредагувавши робочий файл можна виконати синтаксичну перевірку. Для цього виконують команду **Компиляция → Синтаксическая проверка** головного меню «**Компиляция**».



За наявності синтаксичних помилок у коді програма формує звіт про кожну із знайдених помилок у вікні редактора-компілятора.

Виправивши помилки, виконують компіляцію програми в формат *MPM* (*MPM, Microcontroller Program Memory*). Для цього виконують команду **Компиляция → Компилировать в MPM** головного меню «**Компиляция**». Під час компіляції символічний код програми перекладається в машинний код формується однойменний файл з розширенням «*.*MPM*».

Комплекс моделювання *SCM* МК51 за замовчуванням працює як з *MPM*-файлами, так і з файлами формату *HEX* – стандартним форматом подання пам'яті програм (розширення «*.*HEX*»). Для компіляції тексту програми у *HEX*-файл виконують команду **Компиляция → Сгенерировать HEX** головного меню «**Компиляция**».

Для запуску програми виконують команду **Компиляция → Выполнить** головного меню «**Компиляция**», або відповідну команду панелі інструментів.

Виконання програми

Зовнішній вигляд вікна комплексу *SCM* МК51 під час виконання програми зображений на рис. В.4.

Після запуску на виконання програма завантажується в пам’ять програм і відображується в лівій частині головного вікна *SCM* МК51. В правій частині головного вікна комплексу відображується вміст НОЗП та вміст пам’яті даних.

В центральній частині відображається структура та вміст основних функціональних частин МК51: чотири банки регістрів, блок управління, таймери, АЛП та блок послідовного інтерфейсу та переривань. Пристрої поєднані між собою каналами зв’язку. Таким чином, модель МК51, що розміщена у центральній частині головного вікна комплексу відображує структуру та принципи функціонування МК51.

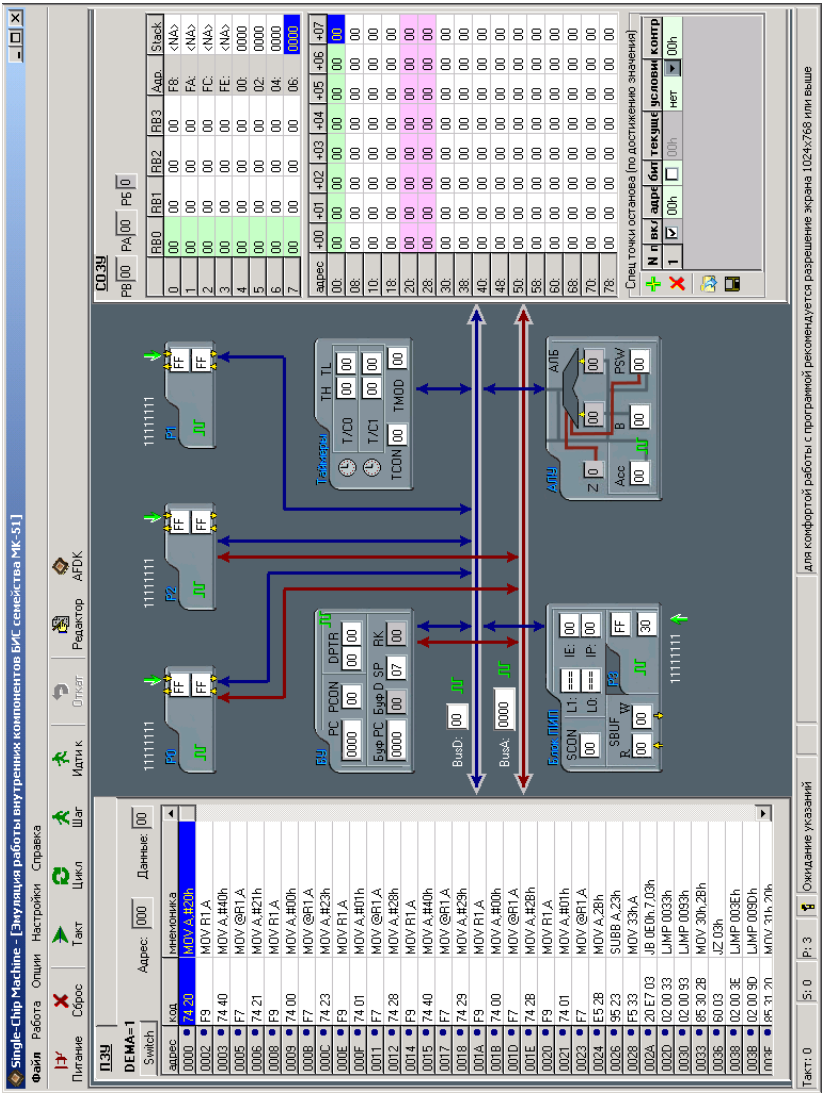
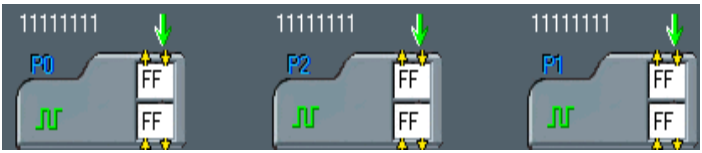


Рис. В.4. Зовнішній вид головного вікна комплексу *SCM* MK51 під час виконання програми

Порти вводу/виводу

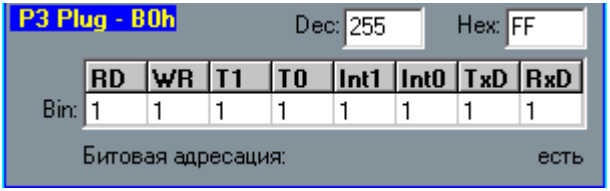


Комплекс моделювання надає можливість роботи з трьома двоспрямованими портами вводу/виводу *P*0, *P*1, *P*2, що забезпечують обмін інформацією з зовнішніми пристроями.

Значення портів встановлюють, навівши мишу на поле для вводу даних потрібного порту. В результаті відкривається наступне вікно редагування буферу порту:

Значення кожного біта порту встановлюють за допомогою подвійного кліка миші.

Кожна лінія порту *P*3 має індивідуальну альтернативну функцію, тому він віднесений на схемі до блоку ПІП:



Моделюючий комплекс дозволяє встановити вихідні значення регістрів всіх функціональних пристроїв МК51 (рис. В.4).

Натискаючи відповідні піктограми можна продивитись діаграми сигналів пристроїв (рис.В.2).

Налагоджування та відпрацьовування програми у комплексі *SCM* МК51 виконується аналогічно комплексу моделювання *SCM* МК48 (Додаток Б).

ДОДАТОК Г

УЧБОВО-НАЛАГОДЖУВАЛЬНИЙ СТЕНД

© Під час практичного засвоєння матеріалу другого модуля «Однокристальний мікроконтроллер КР1816ВЕ51» застосовується учбово-налагоджувальний стенд, розроблений компанією *OPEN SYSTEM*, Україна (*http://opensys.com.ua*). Також компанією представлений даний технічний опис та інструкція до експлуатації, які дозволяють отримати базові уявлення за роботи зі стендом, а також за порядком і правилами його експлуатації, виконання яких забезпечить безперебійну роботу стенду. Окрім того в практичній частині до третього модуля (розділ 8.1) виконуються практичні роботи (Практичні роботи 3.4 – 3.6) представлені компанією *OPEN SYSTEM*.

Г.1. Призначення учбово-налагоджувального стенду

Учбово-налагоджувальний стенд (УНС) «*EV*8031/*AVR*»– програмно-апаратний комплекс, орієнтований на використання в учбових цілях, як засіб розробки програмного забезпечення для контролерів на базі однокристальної ЕОМ серії *MCS*-51 (КР1816ВЕ51), а також на базі контролерів архітектури *AVR*. Загальний вид стенда представлений на рис. Г.1.

© Стенд комплектується восьмирозрядним КМОП мікроконтролером *AT*89*C*51 фірми *ATMEL*. Мікросхема випускаються в стандартному *DIP*-корпусі, запитується напругою 5V ± 20%. Мікроконтролер сумісний за системою команд із широко розповсюдженою мікросхемою фірми *INTEL* - *MCS*-51 [IC8051] (вітчизняний аналог МК1816ВЕ51). Мікроконтролер *AT*89*C*51 містить: 4 Кб *FLASH* ПЗП; 128 байтів ОЗП; тридцять дві лінії вводу/виводу, що програмуються; два шістнадцятирозрядних таймери/лічильника; повнодуплексний послідовний порт (*UART*); п'ять векторних дворівневих переривання; вмонтований генератор; схему формування тактової послідовності. Мікроконтролер *AT*89*C*51 має менші габарити і більш зручний у застосуванні ніж попередній аналог. Застосований в якості пам’яті команд електрично перепрограмуємий ПЗП, дає можливість швидко змінювати програмне забезпечення. Застосування системи команд популярного у свій час мікроконтроллера дозволяє використовувати для створення й налагодження програм уже існуючі інструментальні програмні засоби.

© – коментарі від авторів

Рис. Г.1. Загальний вигляд учбово-налагоджувального стенда *EV8031/AVR*

Г.2. Схема підключення стенду до персонального комп'ютера. Живлення стенду.

Зв'язок учбово-налагоджувального стенду «*EV*8031/*AVR*» з персональним комп'ютером (ПК) здійснюється через *COM*-порт. Схеми підключення УНС до ПК наведені на рис. Г.2 та рис. Г.3. Для підключення застосовується універсальний з'єднувальний кабель з двома рознімачами – 25 *PIN* (*X*3) і 9 *PIN* (*X*4) для з'єднання з одним із *COM*-портів ПК. З'єднувальний кабель має рознімач підключення до комп'ютерного блоку живлення (*X*2) і рознімач для підключення стенду (*X*1).

Увага!!! Для справної і правильної роботи стенду, його необхідно підключати тільки до одного *COM*-порту ПК. Не рекомендується від'єднувати стенд від ПК або від'єднувати плату розширення від стенду за включеного живлення.

Г.3. Технічні характеристики

Технічні характеристики стенду «*EV*8031/*AVR*»:

– використані однокристальні процесори *AT89C51, AT89C52, AT90S8515* (*ATmega8515*) (корпус *DIP-40*);

– пам'ять програм – 16 Кб;

– пам'ять даних – 16 Кб;

– послідовна *EEPROM* пам'ять, 256 байт (*AT24C02*);

– два послідовних канали передачі даних *RS*232;

– системний інтерфейс;

– інтерфейс розширення (шістнадцять вихідних ліній, вісім двоспрямованих ліній (вхід/вихід), порт *P*1 мікроконтролера);

– клавіатура 4×3;

– статична чотирирозрядна семисегментна світлодіодна індикація;

– цифроаналоговий и аналого-цифровий перетворювач (на платі розширення);

– генератор з фіксованою частотою генерації – 10 кГц, генератор із змінною частотою генерації від 1 кГц до 50 кГц (на платі розширення);

– динамічна чотирирозрядна семисегментна індикація (на платі розширення);

– пристрій дискретного вводу інформації: дві кнопки;

– статична світлодіодна індикація, (вісім індикаторів);

знакосинтезуючий світлодіодний індикатор 5×7 (один індикатор на платі розширення).

Перелік комплектуючих мікросхем наведений у табл. Г.2.

Перший варіант підключення

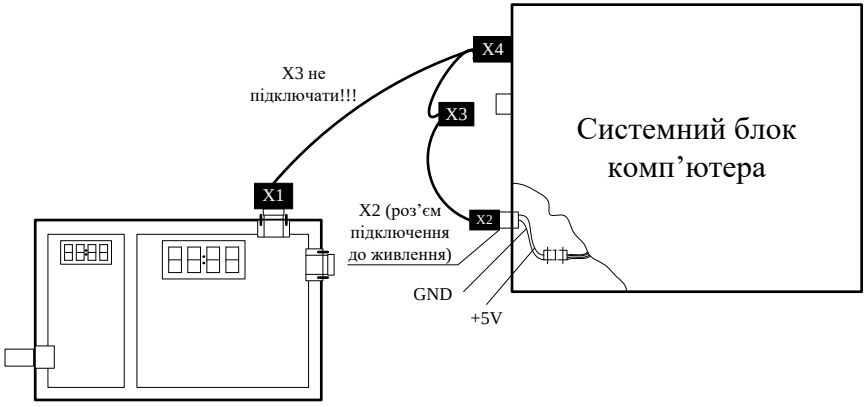


Рис. Г.2. Схема підключення стенду *EV*8031/*AVR* до ПК + живлення

Другий варіант підключення

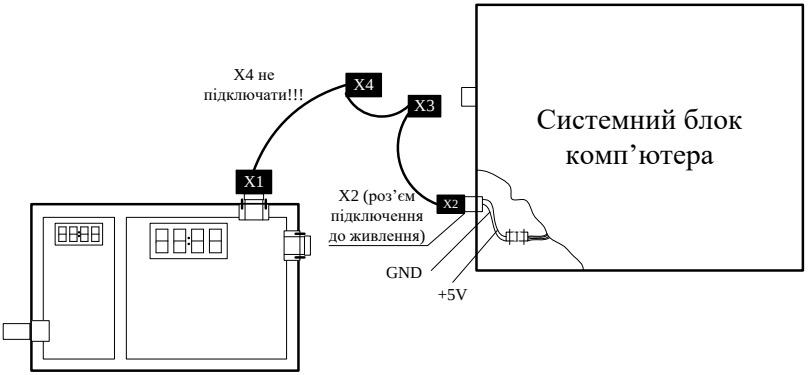


Рис. Г.3. Схема підключення стенду *EV*8031/*AVR* до ПК + живлення.

Г.4. Опис стенду

Структурна схема. Розміщення елементів на платі. Розподіл пам'яті

Структурна схема стенда представлена на рис. Г.4.

Схема розміщення елементів стенду на платі наведена на рис. Г.5. На схемі розміщення застосовуються наступні позначення:

*X*1	–	системний інтерфейс з повним адресним простором;
*X*10	–	інтерфейс розширення для підключення зовнішніх пристроїв з використанням паралельного інтерфейсу;
*X*11	–	інтерфейс послідовного порту *COM*1 для зв'язку стенду з ПК;
*X*12	–	інтерфейс послідовного порту *COM*2 для зв'язку стенду з іншими пристроями, що мають стандартний порт *RS*232*C*;
*X*3	–	інтерфейс програмування *AVR*;
*X*14, *X*15	–	перемичка підключення пристроїв шини I^2C до процесора;

Для завантаження виконуваної на стенді програми використовується програма-завантажувач, що знаходиться в *Flash*-пам'яті мікроконтролера *AT*89*C*51 (*DD1*). Програма-завантажувач проводить ініціалізацію послідовного приймача-передавача мікроконтролера, перевіряє наявність і ємність пам'яті даних.

Пам'ять ОЗП об'ємом 32Кб розподіляється на дві частини ємністю 16Кб. Одна частина використовується для реалізації пам'яті програм, друга – для пам'яті даних.

У режимі завантаження вся пам'ять 32Кб відображується в єдиний адресний простір, як пам'ять даних. Розподіл пам'яті стенду наведений на рис. Г.6.

Під час надходження даних з послідовного порту персонального комп'ютера в послідовний порт (рознімач *X*11) стенду, МК записує їх в частину ОЗП, що відведена під пам'ять програм. Сигнали управління – *PME, WR, RD, ALE*, що формуються процесором і застосовуються для звернення до пам'яті даних поступають через системний контролер. Після підтвердження останнього байта завантажувач формує сигнал запуску програми, шляхом запису управляючого коду в системний контролер.

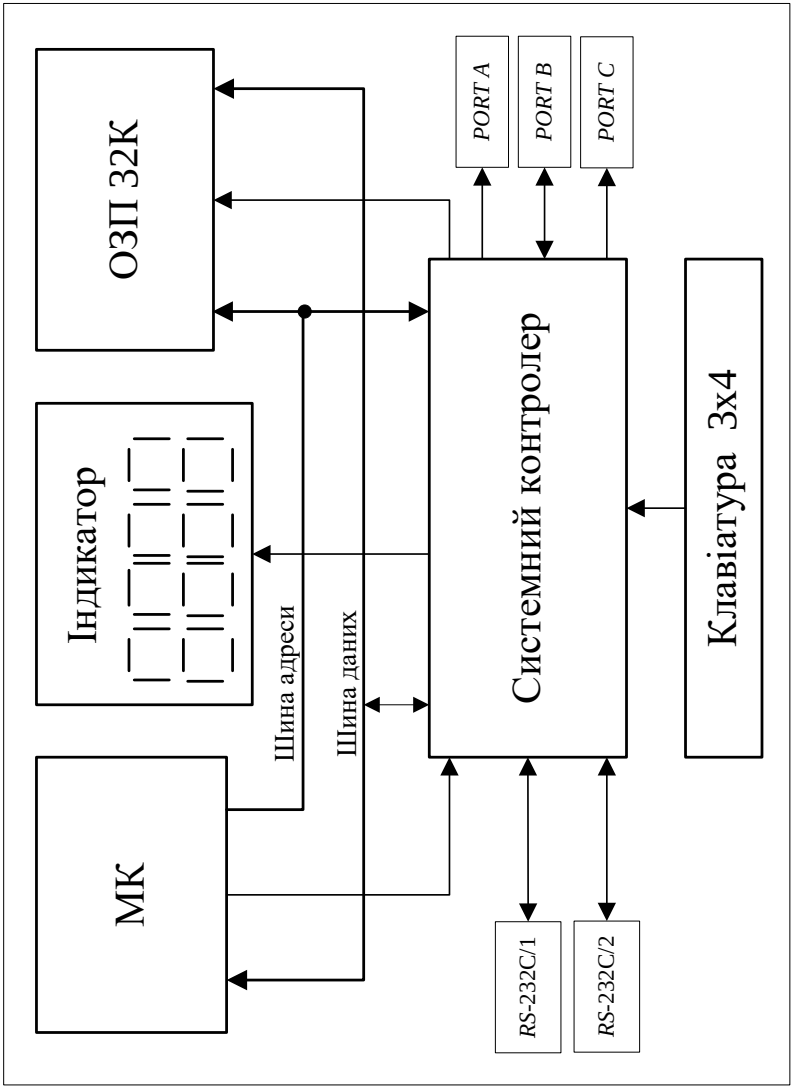


Рис. Г.4. Структурна схема стенду *EV*8031/*AVR*

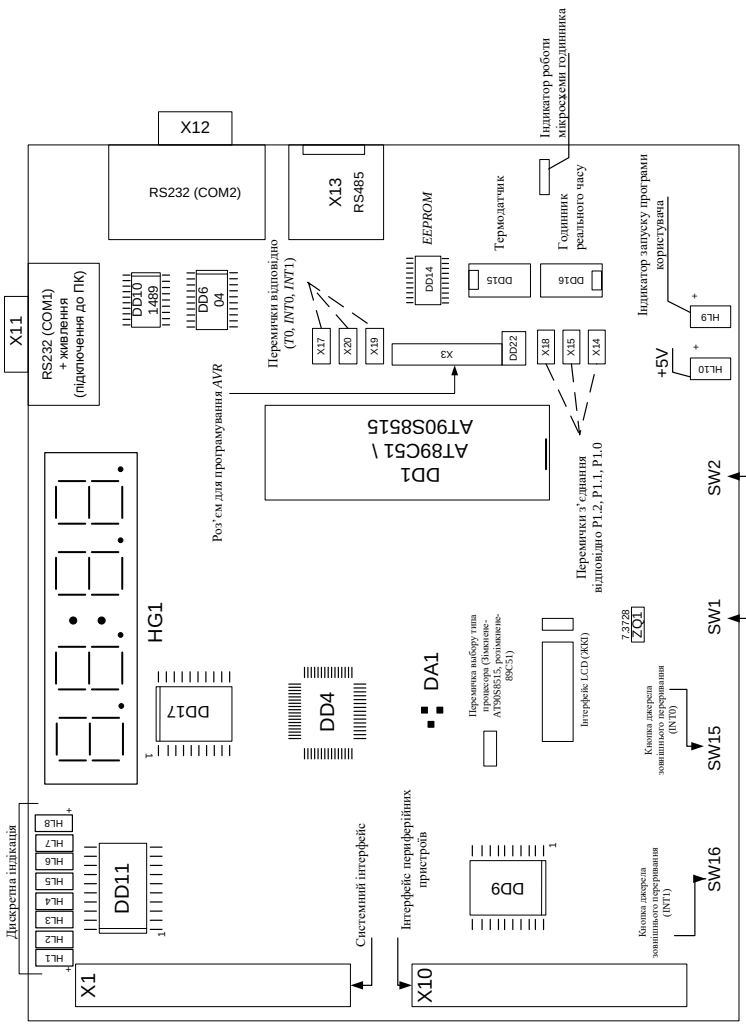


Рис. Г.5. Схема розміщення елементів стенду

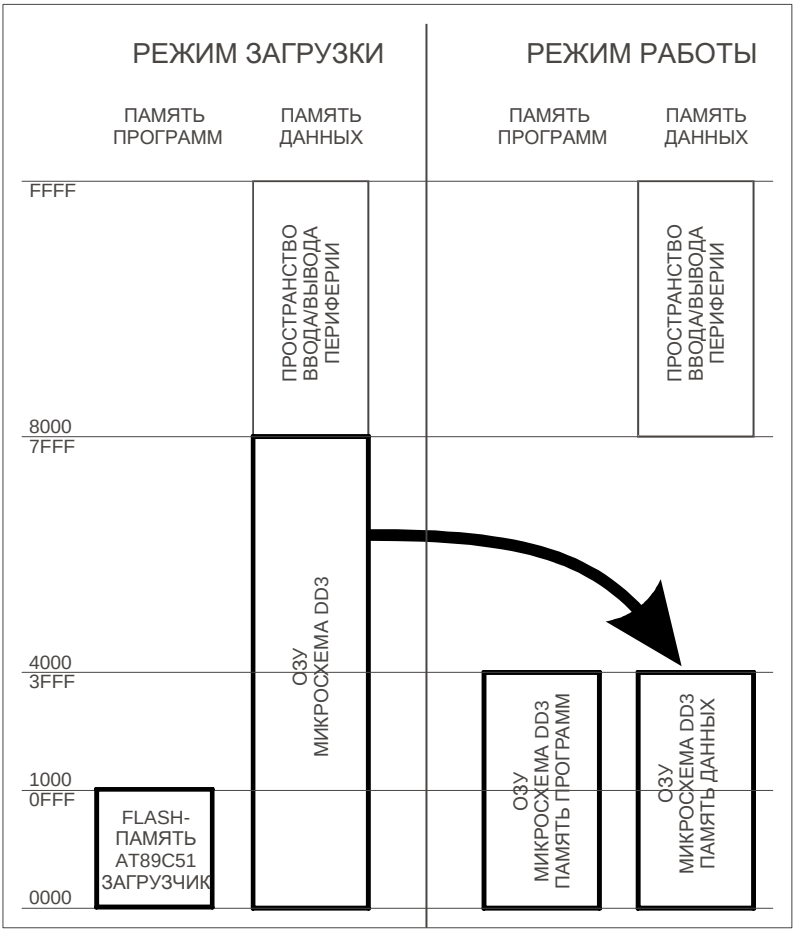


Рис. Г.6. Розподіл пам'яті стенду

© Під час опису структури та принципу функціонування учбово-налагоджувального стенду використовуються позначення застосовані на схемах електричних принципових стенду, які умовно не представлені у даному технічному описі. Принципові схеми для вивчення стенду та виконання практичних робіт можна отримати у викладача в навчальній лабораторії. До кожної з практичних робіт (практичні роботи 3.3 – 3.4), у розділі 6.3 представлені фрагменти схем електричних функціональних, на яких також застосовуються відповідні даному технічному опису позначення.

Кнопка *SW*2, застосовується для формування сигналу скидання на вході *RESET* процесора, тобто переходу стенда в режим завантаження і очікування прийому даних з послідовного порту. Процесор готовий приймати дані в пам'ять даних.

Кнопка *SW*1, необхідна для перезапуску завантаженої з персонального комп'ютера програми, що знаходиться в пам'яті програм (*DD3*). Під час натискання кнопки *SW*1, спалахує світлодіод *HL*9. При цьому можливий новий запис програми в стенд з персонального комп'ютера. Під час передачі даних з персонального комп'ютера в стенд, комп'ютер на виводі *RI* послідовного порту формує сигнал, який через системний контролер скидає процесор, також як і кнопка *SW*2.

Системний контролер управляє режимами роботи, вироблення управляючих сигналів на ОЗУ, регістри клямки, динамічним світлодіодним індикатором, клавіатурою. Вся логіка стенду реалізована на програмованій логічній мікросхемі *EPM*7128*STC*100 (*DD*4).

Адресація процесора до периферійних пристроїв стенду реалізовано як адресація до елементів пам'яті в адресному просторі від 8000*h* до *FFFFh*. Сигнали вибірки периферійних пристроїв формуються дешифратором адреси усередині мікросхеми системного контролера *DD4*.

Карта портів вводу/виводу стенду наведена у табл. Г.1.

Перелік комплектуючих мікросхем наведений у табл. Г.2.

Послідовний приймач-передавач

Модуль послідовного зв'язку сформований на мікросхемі приймача 1489, передавача *74HC04*, мультиплексора каналу передачі (усередині системного контролера).

Швидкість обміну по послідовному порту в режимі завантаження 9600б/с. Швидкість обміну по послідовному порту у налагоджуваній програмі може бути змінена.

Вибір каналу послідовної передачі здійснюється сигналами *CFG*0, *CFG*1 за адресою *9001H*. Установка цих бітів в "логічний нуль" включає порт P1, на схемі *X*11, цей порт має неповний набір сигналів ($R \times D$, $T \times D$, *RI*) і призначений для запису програми в стенд.

Програмна установка сигналів *CFG*0 в "0", а *CFG*1 в «1» формує вибірку додаткового каналу послідовної передачі даних, рознімач *X*12. Додатковий послідовний канал має повний набір сигналів інтерфейсу *RS*-232*C*.

Таблиця Г.1. Карта портів вводу/виводу стенду

Адреса [16]	**Тип циклу**	***B*7**	***B*6**	***B*5**	***B*4**	***B*3**	***B*2**	***B*1**	***B*0**	**Ім'я**
Порти периферійних пристроїв										
8**0	Запис	[Порт *A*]								*PA_REG*
8**1	Запис	[Порт *B*]								*PB_REG*
8**2	Запис	[Порт *C*]								*PC_REG*
8**3	Запис	*	*	*	*	*	*TRISC*	*	*	*TRIS*
ЖКІ										
8**4	Запис	Регістр команд ЖК індикатора								*LCD_CMD*
8**5	Запис	Регістр даних ЖК індикатора								*LCD_DATA*
Послідовний порт										
9***	Читання	*CTS*	*DSR*	*DCD*	*RI*	*KL*3	*KL*2	*KL*1	*KL*0	*US_REG*
*C**0*	Запис	*	*	*	*	*DTR*	*RTS*	*CFG*1	*CFG*0	*UC_REG*
*CFG*1	*CFG*0									
0	0	*RS*-232				*COM1, X*11				

Продовження табл. Г.1.

0	1	*RS*-232				*COM2, X*12				
1	0	*RS*-485				Прийом, *X*13				
1	1	*RS*-485				Передача, *X*13				
Індикатор і світлодіоди										
*A***0	Запис	[Регістр індикатора 0]								*DISPLAY*[0]
*A***1	Запис	[Регістр індикатора 1]								*DISPLAY*[1]
*A***2	Запис	<зарезервовано>								*DISPLAY*[2]
*A***3	Запис	<зарезервовано>								*DISPLAY*[3]
*A***4	Запис	*DP3*	*DP*2	*DP*1	*DP*0	*BL*3	*BL*2	*BL*1	*BL*0	*DC_REG*
*A***5	Запис	<зарезервовано>								*EDC_REG*
*A***6	Запис	*LED*7	*LED*6	*LED*5	*LED*4	*LED*3	*LED*2	*LED*1	*LED*0	*LED_REG*
Управління роботою										
*A***7	Запис	*	*	*	*	*	*	*	*RUN*	*SYS_CTL*
Сумісні регістри										
*B***0	Запис	[Регістр індикатора 1]								*DISPLAYB*

* – довільне значення в тетраді, числа від 0 до *Fh*

Таблиця Г.2. Перелік комплектуючих мікросхем

№	Позначення на схемі	Назва мікросхеми	Вітчизняний аналог (найближчий)	Стислий опис
1	*DD*1	*AT*89*C*52 *AT*89*S*52	немає (МК1816ВЕ51)	однокристальна ЕОМ
2	*DD*2, *DD*7, *DD*9, *DD*11, *DD*17	74*HC*573*N*	1533 ИР33	восьмирозрядний регістр
3	*DD*3	62256	немає (К537РУ17)	статичне ОЗП 32Кб
4	*DD*4	*EPM*7128*STC*100	немає	програмована логічна схема
5	*DD*18	74*HC*04	К1564ЛН2	шість КМОП інверторів
6	*DD*10	1489	К559ИП20	перетворювач рівня *RS*-232*C*
7	*DD*12	*ADM*485	невідомий	перетворювач рівня *RS*-485
8	*DD*14	*AT*29*C*02	немає	ЕСПЗП 2 Кбит
9	*DD*15	*DS*1621	немає	цифровий температурний датчик
10	*DD*16	*DS*1302	немає	годинник реального часу
11	*DD*23, *DD*24	*ADM*202	немає	перетворювач рівня *RS*-232*C*

Світлодіодний індикатор

Чотирирозрядний семисегментний світлодіодний індикатор підключений до системного контролера, який автоматично виконує динамічну регенерацію і декодування двійкового коду в код семисегментного індикатора. Індикатор працює завжди, відразу після подачі живлення. Контролер індикатора містить два восьмирозрядних регістри, вміст яких відображається на індикаторі. Вміст регістра з адресою 0**A*000 відображається на двох лівих розрядах, вміст регістра з адресою 0**A*001*(*0**B*000*)* – на двох правих розрядах в шістнадцятирічній формі. Управління десятковими крапками і гасінням здійснюється через регістр *DC_REG(*0**A*004*)*. Біти [*DP*3..*DP*0] управляють десятковими крапками. Запис одиниці у відповідний розряд включає десяткову крапку. Біти [*BL*3..*BL*0] управляють гасінням розрядів індикатора. Запис одиниці в ці біти викликає гасіння відповідного розряду індикатора.

Матрична клавіатура

Стан стовпця матриці клавіатури прочитується з комірки з базовою адресою *0*9000*, біти [3..0]. Відповідний стовпець вибирається нулем в розрядах адреси [*A*2..*A*0]. Тобто, адреса *0*9006* вибирає перший стовпець, адреса *0*9005* – другий стовпець, адреса *0*9003* – третій стовпець. Ознака натиснутої кнопки прочитується як нуль у відповідному розряді.

Включення портів ОЕВМ і EEPROM пам'яті

Виводи *P*1.0, *P*1.1 МК можуть бути відключені від внутрішньої периферії стенду (шина I^2C) перемичками *X*14, *X*15. На рознімач інтерфейсу розширення сигнали приходять, минувши перемички.

Г.5. Опис плат розширення

Призначення

Плата розширення призначена для проведення лабораторних робіт, зв'язаних з аналого-цифрового і частотного перетворення, а також з обробкою дискретних сигналів. Структурна схема плати розширення приведена на рис. Г.7. На структурній схемі застосовані наступні позначення:

8888 – чотирирозрядна динамічна індикація;
ІПП – інтерфейс периферійних пристроїв;
ЦАП – цифроаналоговий перетворювач;

СДІ – світлодіодні індикатори;
ЗСІ – знакосинтезуючий індикатор 5х7;
ГПЧ – генератор із змінною частотою генерації;
INT – кнопка запиту переривання;
ДВН – джерело вимірюваної напруги;
ШД – шина даних.

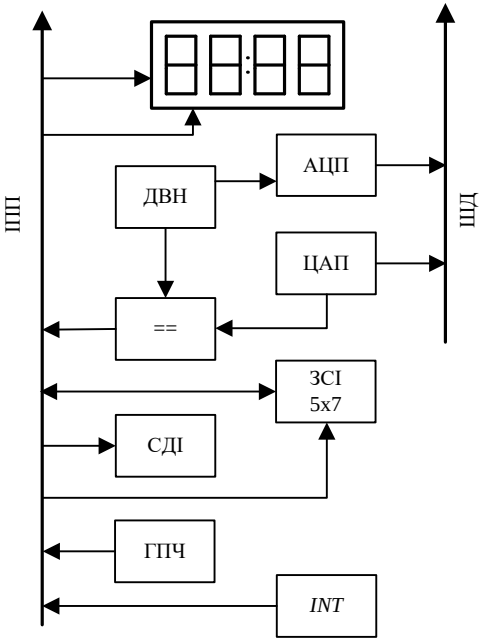


Рис. Г.7. Структурна схема плати розширення

Цифроаналоговий перетворювач

Цифроаналоговий перетворювач (ЦАП) виконаний на мікросхемі *AD*7801 *DD*2 (восьмирозрядний ЦАП). Вхідними сигналами для ЦАП є виводи *AD*7 – *AD*0. Вихідний сигнал знімається з рознімач *BNC*.

Аналого-цифровий перетворювач

Аналого-цифровий перетворювач (ЦАП) виконаний на мікросхемі аналого-цифровий перетворювач *AD*7801, операційному підсилювачі, використовуваним як компаратор *LM*358 *DA*1. Вхідним аналоговим сигналом для АЦП є сигнал із змінного резистора *R*19. Виводи *AD*7 – *AD*0 використовуються для формування цифрового

вхідного коду. На виході ЦАП формується напруга, пропорційна вхідному коду. Сигнал спрацьовування компаратора знімається з *DA*1 – *DA*2 вхід МК *P*1.7. Спрацьовування компаратора візуально видно по загорянню світлодіода *HL*1. Якщо на *P*1.7 «0» світлодіода світиться.

Генератори

У схемі присутній генератор із змінною частотою генерації ~ 1 – 50 Кгц, елементи *R*1, *R*4, *R*5, *R*7, *R*10, *R*11, *R*15, *R*16, *C*3, *VT*1, *DA*1–1 (зміна частоти здійснюється за допомогою резистора *R*4), і генератор з фіксованою частотою ~ 10 Кгц, елементи *R*19, *R*20, *C*16, *DD*18–1, *DD*18–2, *DD*18–3.

Виведення дискретної інформації

Виведення дискретної інформації здійснюється за допомогою чотирьох розрядного семисегментного індикатора *HL*2 включеного по схемі динамічної індикації. Управління динамічною індикацією здійснюється за допомогою елементів *DD*3 (лінія даних *A, B, C, D, E, F, G, DP -PB*0, *PB*1, *PB*2, *PB*3, *PB*4, *PB*5, *PB*6, *PB*7) сигнали поступають з порту PB, сигнали вибірки відповідного індикатора поступають від ліній *PC*0, *PC*1 порту *C*.

Схема розташування елементів плати розширення зображена на рис. Г.8. На схемі розташування застосовані наступні позначення:

*HG*1	–	знакосинтезуючий індикатор 5×7;
*HL*2	–	чотирирозрядна динамічна індикація;
*HL*1	–	світлодіодний індикатор спрацьовування компаратора;
*J*1	–	перемичка підключення до рознімач *J*2 виходу генератора з постійною частотою генерації;
*J*2	–	рознімач підключення зовнішніх контрольно-вимірювальних приладів;
*J*3	–	перемичка підключення до рознімач *J*2 виходу генератора із змінною частотою генерації;
*J*4	–	перемичка підключення до рознімач виходу АЦП;
*J*5	–	підключення як джерело зовнішнього переривання *INT*1 кнопки *S*11;

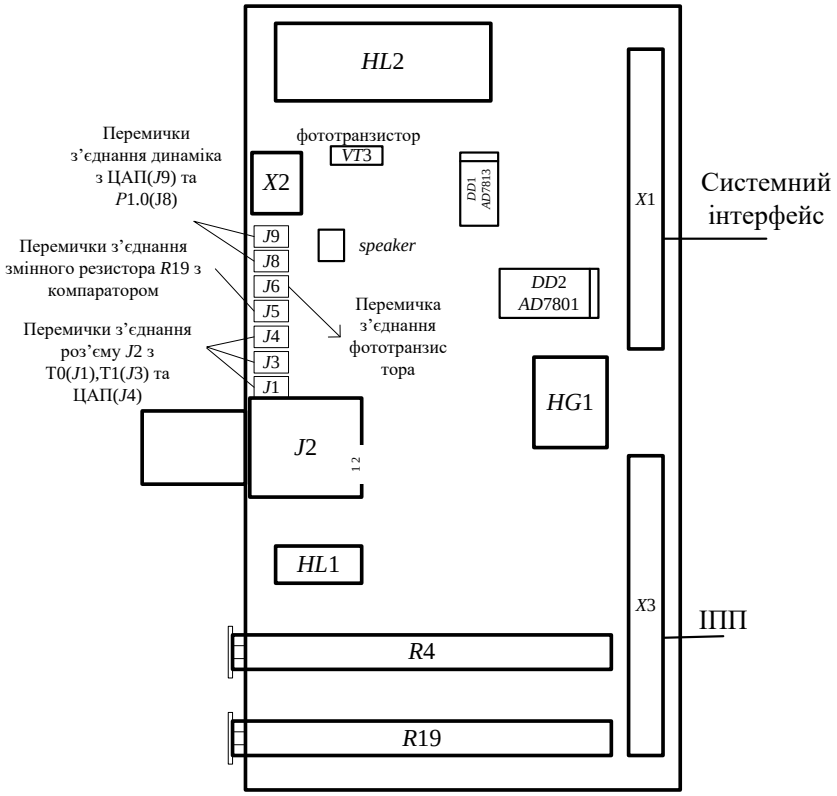


Рис. Г.8 Схема розташування елементів плати розширення

*J*6	– підключення як джерело зовнішнього переривання *INT*1 зовнішнього джерела, яке може бути підключений через рознімач *JP*1;
J7	інтерфейс підключення плати розширення до стенду;
*J*8	підключення до входу АЦП зовнішнього джерела сигналу, підключеного до рознімача *JP*2;
*J*9	підключення як джерело сигналу для АЦП змінного резистора *R*27;
*R*19	змінний резистор, джерело вхідного сигналу для АЦП;
*R*4	змінний резистор, змінює частоту генерації генератора імпульсів;

Плата розширення для систем автоматичного управління

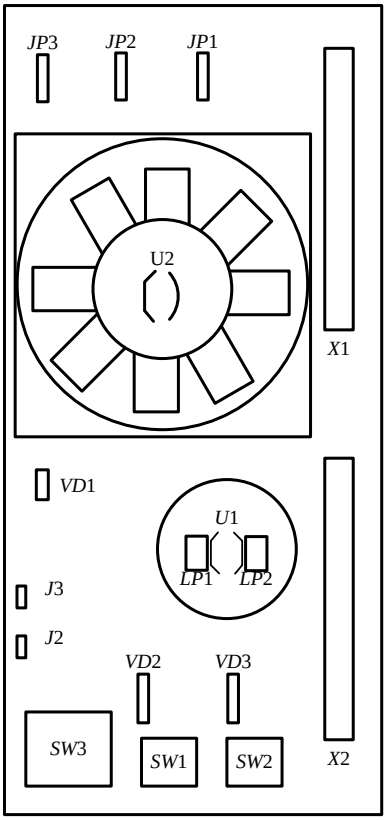


Рис. Г.9. Схема розташування елементів плати розширення для систем автоматичного управління

Схема розташування елементів плати розширення зображена на рис. Г.8. На схемі розташування застосовані наступні позначення:

*JP*1	перемичка вибору виду регулювання. Вибирається регулювання частоти оборотів двигуна (*MOTOR*), або регулювання температури (*LAMP*);
*JP*2	перемичка вибору способу регулювання. Вибирається регулювання лінійне, змінюючи амплітуду вихідного сигналу з ЦАП (*LINE*), або регулювання ШІМ (широтно-імпульсна модуляція) (*PWM*);
*JP*3	перемичка вибору об'єкту регулювання. Вибирається

	режим регулювання частоти обертання/температури (*REG*), або режим управління звуком (*SOUND*);
*J*2, *J*3	перемички вибору способу управління звуком: *J*2 замкнута, *J*3 розімкнена – звуком управляють безпосередньо в мікроконтролері; *J*2 розімкнена, *J*3 замкнута – звуком управляють за допомогою ЦАП;
*X*1, *X*3	рознімач підключення до основної плати стенду;
*U*1	датчик температури;
*U*2	датчик оборотів (на основі ефекту Холу);
*LP*1, *LP*2	нагрівальні елементи (лампи накалювання);
*SW*1, *SW*2	дискретні кнопки;
*SW*3	багатооборотний перемикач;
*VD*1	індикатор обертання двигуна;
*VD*2, *VD*3	індикатори натискання кнопок *SW*1, *SW*2.

Г.6. Схема з'єднувальних кабелів

Для підключення учбово-налагоджувального стенду до персонального комп'ютера використовується комплект кабелів, схема з'єднання яких зображена на рис. Г.10.

Увага! Оскільки до складу даного кабелю входять лінії живлення +5В. Категорично забороняється підключати цей кабель до інших пристроїв.

Г.7. Інтерфейси

Послідовні порти

Інтерфейс послідовного порту *COM*1 *RS*232 призначений для зв'язку стенду з персональним комп'ютером, прийому даних, що завантажуються в пам'ять програм стенду.

Таблиця 6.2. Інтерфейс послідовного порту *COM*1

Номер контакту	**Найменування**	**Номер контакту**	**Найменування**
3	$R \times D$	9	*RI*
2	$T \times D$	5	*GND*

Інтерфейс послідовного порту *COM*2 *RS*232 призначений для зв'язку стенду з будь-яким пристроєм, що має стандартний

послідовний порт *RS*232 (персональний комп'ютер, принтер, плоттер і таке інше).

Таблиця 6.2. Інтерфейс послідовного порту *COM*2

Номер контакту	Найменування	Номер контакту	Найменування
1	*DCD*	6	*DSR*
2	$R \times D$	7	*RTS*
3	$T \times D$	8	*CTS*
4	*DTR*	9	*RI*
5	*GND*		

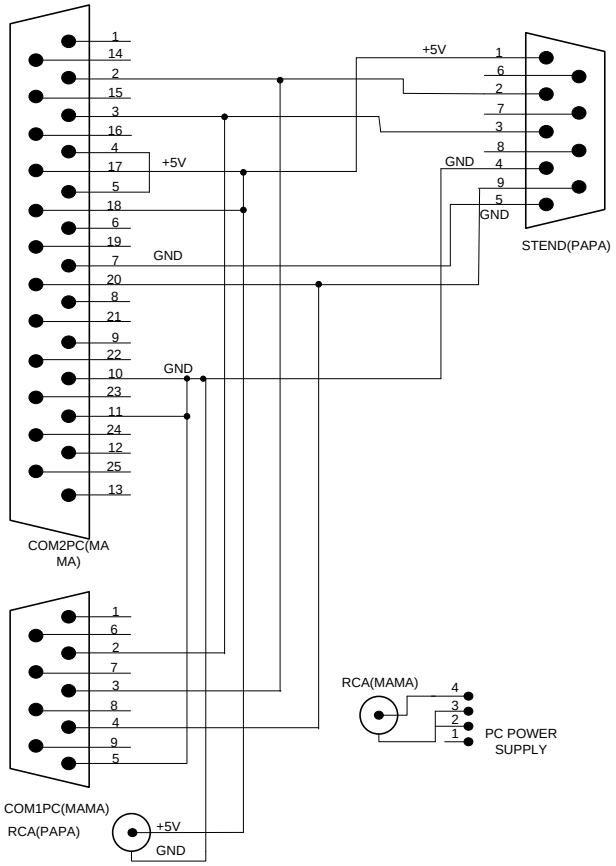


Рис. Г.10. Схема з'єднувальних кабелів

Системний інтерфейс

Системний інтерфейс призначений для підключення зовнішніх пристроїв, що розробляються. Наявність повного адресного простору сигналів *ALE, PME, WR, RD* дозволяє здійснити вибірку практично будь-якого пристрою. Живлення модулів, що підключаються, проводиться від стенду. Вільні лінії на інтерфейсі 23 – 26 дозволяють використання сигналів *CSI, CS*4 – *CS*7, при відповідному підключенні (табл. 6.3).

Таблиця 6.3. Системний інтерфейс

Номер контакту	**Найменування**	**Номер контакту**	**Найменування**	**Номер контакту**	**Найменування**
1	*GND*	16	*A*9	31	*AD*0
2	*GND*	17	*A*10	32	*AD*1
3	*INT*0	18	*A*11	33	*AD*2
4	*INT*1	19	*A*12	34	*AD*3
5	*T*0	20	*A*13	35	*AD*4
6	*T*1	21	*A*14	36	*AD*5
7	*A*0	22	*A*15	37	*AD*6
8	*A*1	23	*CS*6	38	*AD*7
9	*A*2	24	*CS*7	39	*UCC*
10	*A*3	25	$R \times D$	40	*UCC*
11	*A*4	26	$T \times D$		
12	*A*5	27	*RD*		
13	*A6*	28	*WR*		
14	*A7*	29	*ALE*		
15	*A8*	30	*PME*		

Інтерфейс розширення

Інтерфейс розширення призначений для підключення зовнішніх пристроїв, що розробляються (табл. 6.3).

Наявність в обох інтерфейсах сигналів *INT*0, *INT*1, *T*0, *T*1 дозволяє використовувати в пристроях, що розробляються, апаратні елементи системи переривання мікропроцесора.

Таблиця 6.3. Інтерфейс розширення

Номер контакту	Найменування	Номер контакту	Найменування	Номер контакту	Найменування
1	*GND*	16	*PC*4	31	*P*1.4
2	*GND*	17	*PC*7	32	*P*1.5
3	*PB*1	18	*PC*6	33	*P*1.6
4	*PB*0	19	*PA*7	34	*P*1.7
5	*PB*3	20	*PA6*	35	*INT*0
6	*PB*2	21	*PA*5	36	*INT*1
7	*PB*5	22	*PA*4	37	*T*0
8	*PB*4	23	*PA*0	38	*T*1
9	*PB*7	24	*PA*1	39	*UCC*
10	*PB*6	25	*PA*2	40	*UCC*
11	*PC*1	26	*PA*3		
12	*PC*0	27	*P*1.0		
13	*PC*3	28	*P*1.1		
14	*PC*2	29	*P*1.2		
15	*PC*5	30	*P*1.3		

Інтерфейс програмування AVR

Інтерфейс призначений для програмування мікропроцесорів сімейства *AVR* (табл. 6.4).

Таблиця 6.4. Інтерфейс програмування *AVR*

Номер контакту	Найменування
1	*MOSI*
2	*MISO*
3	*SCK*
4	*RST*
5	*GND*
6	*VCC*

Г.9. Тестування стенду

Під час подачі напруги на стенд, процесор автоматично визначає розмір пам'яті даних з відображенням на індикаторі стенду *HG*1 числа місткості пам'яті в кіло бітах. (Тестування процесора, регістра клямки, дешифратора адреси, схеми скидання процесора).

Програма *test*1.*hex* дозволяє перевірити канал послідовної передачі даних з персонального комп'ютера на стенд (мікросхема приймача даних) схеми підключення і мультиплексора вибірки, дешифратора адреси, всі розряди елементів статичної індикації.

Тестування плати розширення здійснюється за допомогою вбудованого в ПЗП завантажувача програмою тестування.

Вхід в тестовий режим: утримуючи будь-яку кнопку на клавіатурі натиснути і відпустити кнопку скидання (*SW*2).

Вихід з режиму тестування: натиснути кнопку скидання або за кодом виходу.

Для виклику тесту: ввести номер тесту і натиснути кнопку («#»), ***для виходу з поточного тесту***: натиснути будь-яку кнопку на клавіатурі.

Коди вбудованих тестів:

«1» – вимірювання частоти генератора з незмінною частотою генерації, кГц;

«2» – вимірювання частоти генератора із змінною частотою генерації, кГц;

«3» – «вогник, що біжить» на світлодіодах;

«4» – послідовне засвічування сегментів семисегментних індикаторів;

«5» – «вогник, що біжить» на матриці світлодіодів;

«6» – програма АЦП, відображає десятковий код ЦАП;

«10» – програма тестування мікросхеми годинника реального часу *DS*1302, настройка годинника за допомогою кнопок *SW*15 *і SW*16 розташованих на цифровій платі;

«11» – програма тестування інтегрального датчика температури *DS*1631.

Г.10. Вказівки по техніці безпеки

Учбовий-налагоджувальний стенд «*EV*8031/*AVR*» розрахований на спільну роботу з персональним комп'ютером. Категорично забороняється установка і зняття рознімачів переходників та рознімачів зв'язку при включеному персональному комп'ютері.

Учбовий-налагоджувальний стенд «*EV*8031/*AVR*» не містить напруг небезпечних для життя.

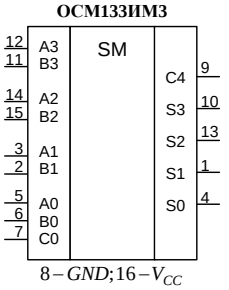
ДОДАТОК Д

ПРИКЛАДИ ЕЛЕМЕНТІВ ЦИФРОВОЇ ТЕХНІКИ

На різних етапах розвитку цифрової схемотехніки промисловістю випускались інтегральні схеми різного ступеню інтеграції від малих ІС до сучасних великих і надвеликих ІС. Для ознайомлення наведені приклади і УГП мікросхем, що мають різний ступень інтеграції.

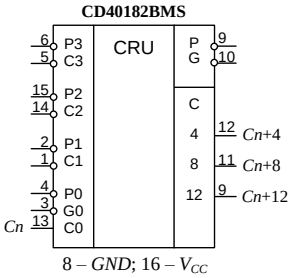
Чотирирозрядний суматор

К555ИМ6 Росія (90 – 97)
ОСМ133ИМ3 Росія
[SN74283] *Texas Instruments Inc*



Пристрій переносу для обслуговування чотирьох секцій АЛУ

1804ВР1, 1533ИП4 Росія (90 – 97)
[CD40182BMS] *Intersil Corp.*



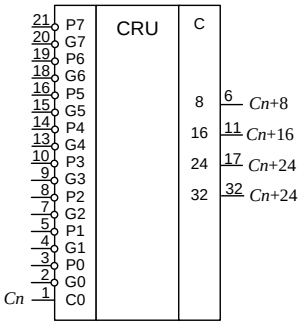
Пристрій переносу для обслуговування восьми секцій АЛУ

[SN74AS882A] *Texas Instruments Inc*

Чотирирозрядний АЛП із скороченою кількістю операцій

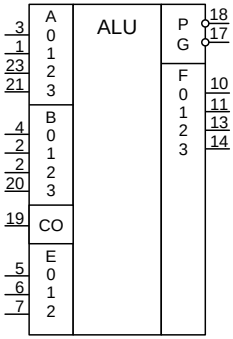
К531ИК2 Росія (90 – 97)
[SN74F382] *Texas Instruments Inc*

SN74AS882A



12 – *GND*; 12 – *GND*; 23 – *NC*

SN74F382

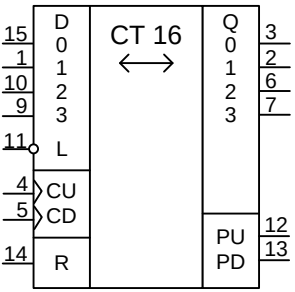


12 – *GND*; 24 – V_{CC}

Чотирирозрядний реверсивний двійковий лічильник

К555ИЕ7 Росія (90 –97)
[SN74LS193] *Texas Instruments Inc*

SN74LS193



8 – *GND*; 16 – V_{CC}

$\overline{L} = 1$ – підраховування,
$\overline{L} = 1$ – завантаження

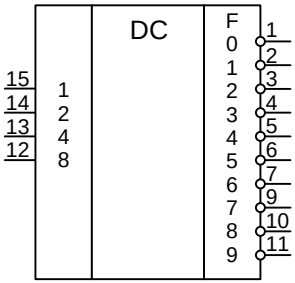
Схема порівняння чотирирозрядних двійкових чисел

К555СП1 Росія (90 – 97)
[DM74LS85N] *Fairchild*
[SN74LS85] *Texas Instruments Inc*

Дешифратор

КМ555ИД6 Росія (90 – 97)
[SN74LS42] *Texas Instruments Inc*

SN74LS42

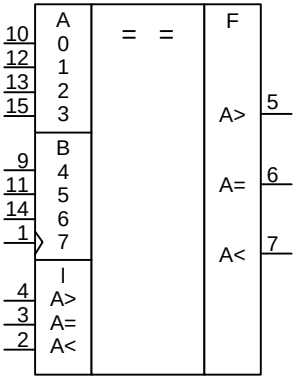


$8 - GND; 16 - V_{CC}$

Восьмиканальний мультиплексор-демультиплексор $8 \to 1/1 \to 8$

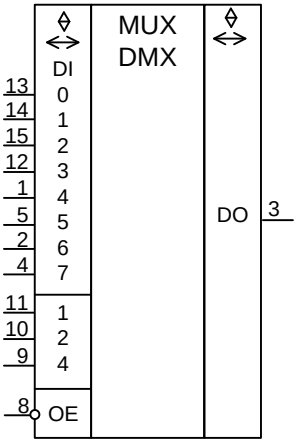
К561КП2 Росія (2003 – 2008)
[CD4051BCN_NL] *Fairchild*

SN74LS85



8 – *GND*; 16 – V_{CC}

К561КП2 [IW4051BN]

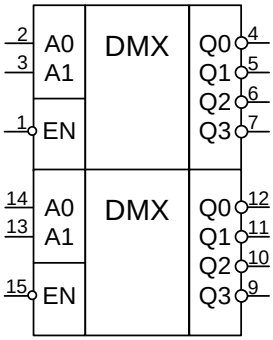


8 – *GND*; 16 – V_{DD}; 7 – V_{EE}

Два декодера-демультиплексора $1 \rightarrow 4$

[74AC139] *Fairchild*

74AC139



8 – *GND*; 16 – V_{DD}

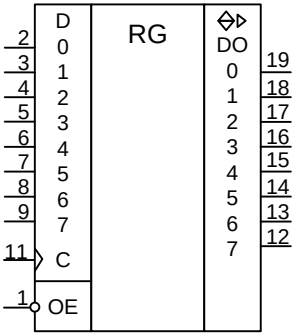
Восьмирозрядний реверсивний зсуваючий регістр

К155ИР13 Росія (90 – 97)
[SN54198] *Texas Instruments Inc*

Восьмирозрядний регістр пам’яті

КР1533ИР33 Росія (2002 – 2007)
[SN74ALS573CN] *Texas Instruments*

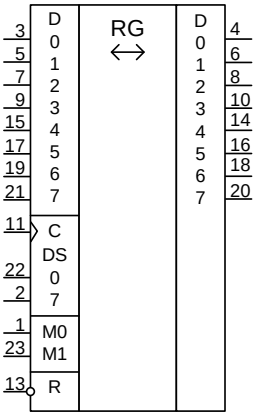
К1533ИР33 [SN74ALS573CN]



10 – *GND*; 20 – V_{CC}

L = 1 та *OE* = 1 виходи $DO_n = D_n$ – режим прямої передачі значень вхідних сигналів на виходи регістру.

SN54198



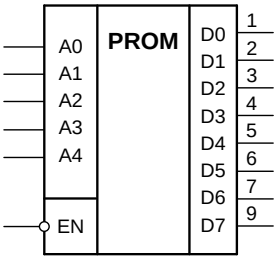
12 – *GND*; 24 – V_{CC}

*M*0=0, *M*1=1 – зсув в сторону старших розрядів; *M*0=1, *M*1=0 – зсув в сторону молодших розрядів; *M*0=0, *M*1=0 – режим зберігання даних.

ППЗП (32 × 8)

[*DM54S188*] *National Semicondactor*

DM54S188

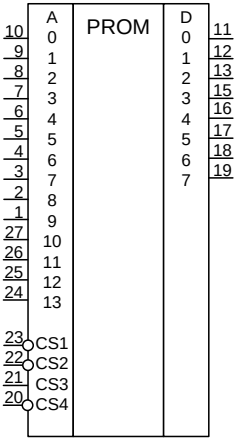


8 – *GND*; 16 – V_{CC}

Високошвидкісний ППЗП (16К × 8)

[*WS57C51C*] *ST Microelectronics*

WS57C51C

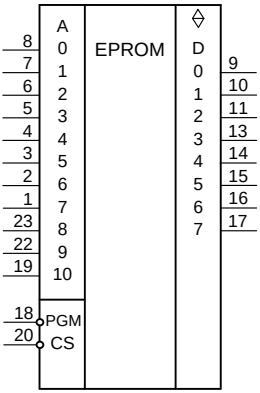


14 – *GND*; 28 – V_{CC}

ЕППЗП (2К × 8)

КР537РQ2 Росія (90 – 97)
[*HN48016P*] *HARRIS Semiconductor*

HN48016P

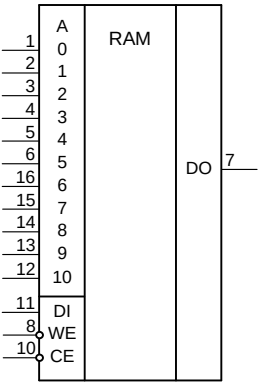


12 – *GND*; 24 – V_{CC}; 21 – V_{PP}

Статичний ОЗП (4К × 1)

КР537РУ2 Росія (90 – 97)
*HM*6504-3 *HARRIS Semiconductor*

HM6504-3

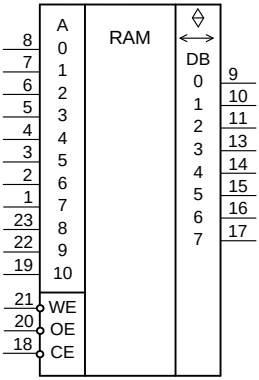


9 – *GND*; 18 – V_{DD}

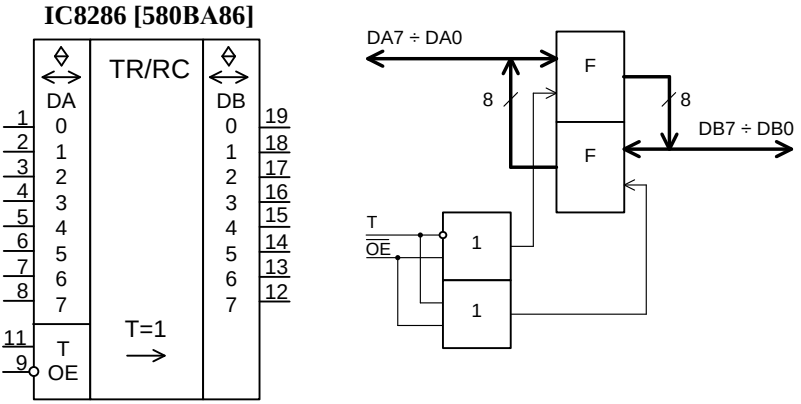
Статичний ОЗП (2К × 8)

КР537РУ10 Росія (90 – 97)
[*HM*6516-9] *National Semiconductor*

***HM*6516-9**



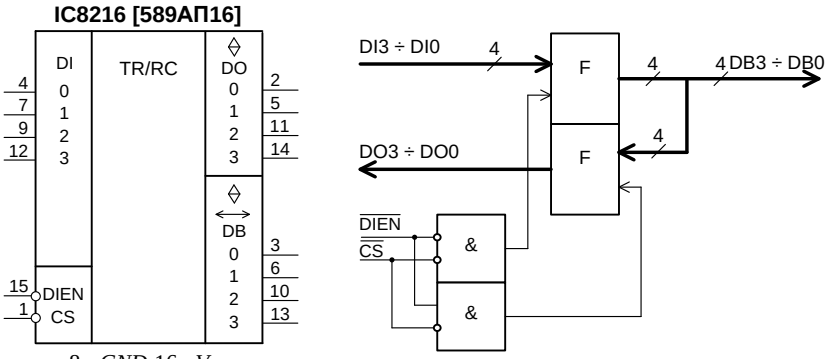
Двоспрямований восьмирозрядний шинний формувач



10 – *GND*; 24 – V_{CC}

DA, *DB* – інформаційні шини, *OE* – підключення формувача, *T* – управління напрямом передачі. За $\overline{OE} = 0$, $T = 1$ – напрям передачі $DA \rightarrow DB$; $\overline{OE} = 0$, $T = 0$ – напрям передачі $DB \rightarrow DA$; $\overline{OE} = 1$, $T = *$ – шинний формувач вимкнено.

Двоспрямований чотирирозрядний шинний формувач



8 – *GND*, 16 – V_{CC}

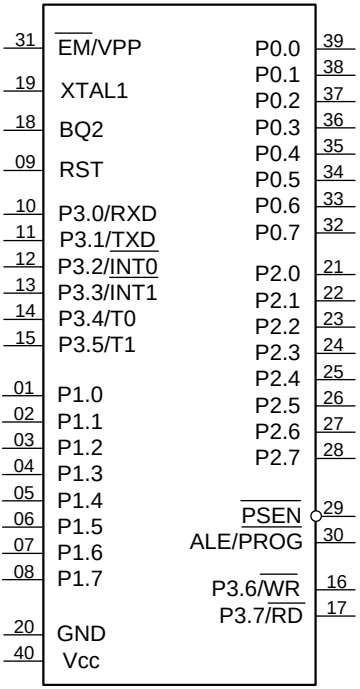
CS – вибір кристалу (підключення мікросхеми); *DI* – вхідні дані; вихідні дані; *DB* – двоспрямовані виводи; *DIEN* – управління напрямом передачі даних ($\overline{DIEN} = 0$ – ввід даних, $\overline{DIEN} = 1$ – вивід даних)

8-бітний мікроконтролер (Аналог К1816КР51)

AT89C51 Atmel

AT90S8515 (ATmega8515) Atmel

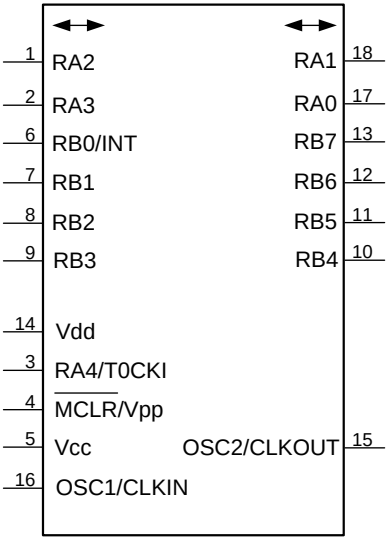
AT89C51/AT90S8515



RISK мікроконтролер

Microchip Technology Inc.

PIC16C71/PIC16F84



ДОДАТОК Є

ЗРАЗКИ ДОКУМЕНТІВ ДЛЯ ОФОРМЛЕННЯ КУРСОВОГО ПРОЕКТУ

Є.1. Зразок оформлення титульного аркуша курсового проекту

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЗВА НАВЧАЛЬНОГО ЗАКЛАДУ

Назва кафедри

КУРСОВИЙ ПРОЕКТ
з дисципліни
НАЗВА ДИСЦИПЛНИ

Виконав_____
Група _____ Спеціальність_____
Залікова книжка № _____

_____
(допущений до захисту)
_____
(підпис викладача)
_____
(захистив з оцінкою)

2009

Є.2. Форма індивідуального завдання

Індивідуальне завдання
до виконання курсового проекту
з дисципліни “НАЗВА ДИСЦИПЛІНИ”

Студента_____
Групи_____

Вихідні дані до розробки	
Вибір елементної бази	
Організація шини	
Вибір системи команд	
КПП. КПДП	
Спосіб множення та розрядність операндів	
Додаткові порти	
Кількість ЗУ	
Функція	
Адреси для ВВ55	
Адреси для інтерфейсу зовнішнього пристрою	
Пам'ять програм	
Пам'ять даних	
Модуль ОП	
Функціональна схема	
Принципова схема	

Завдання видав “_____”
_____
Завдання отримав “_____”
_____

Є.3. Основні написи

Основний напис для перших аркушів креслень і схем

					НАУ 07 4602 003 Е2			
						Літера	Маса	Масштаб
Зм	Лит	№ докум.	Підпис	Дата	Обчислювальний пристрій з мікропрограмним управлінням *Схема електрична функціональна*			
Виконав		Іванов І. І.				Аркуш 1	Аркушів 1	
Керівник		Пєтров П. П.						
Консульт.								
						ФКС 106 6.091501		
Н. контр.								
Зав.каф.		Пєтров П.П.						

Основний напис для перших аркушів текстових документів

					НТУУ КПІ 07 4602 004 ПЗ			
Зм	Лит	№ докум.	Підпис	Дата		Літера	Аркуш	Аркушів
Виконав		Іванов І.І.			Обчислювальний пристрій з мікропрограмним управлінням *Пояснювальна записка*		12	28
Керівник		Пєтров П.П.				ФКС 106 6.091501		
Н.контр.								
Зав.каф.		Пєтров П.П.						

Основний напис для наступних аркушів всіх конструкторських документів

					НАУ 06 4602 003 ПЗ	Аркуш
Зм	Лит	№ докум.	Підпис	Дата		23

Навчальне видання

МІКРОПРОЦЕСОРНІ СИСТЕМИ

Навчальний посібник

Укладачі: ЖАБІН Валерій Іванович
ЖУКОВ Ігор Анатолійович
ТКАЧЕНКО Валентина Василівна
. КЛИМЕНКО Ірина Анатоліївна

Підписано до друку . Формат 60x84x16. Папір офсетний.
Офсетний друк.Ум.фарбовідб.13.Ум. друк.арк.30,75.Обл.-вид.арк.3,5.
Тираж 500 прим. Замовлення № Вид. № 37.III.

Видавництво НАУ
03680. Київ-680, просп. Космонавта Комарова, 1

Свідоцтво про внесення до державного реєстру ДК № 977 від 05.07.2002