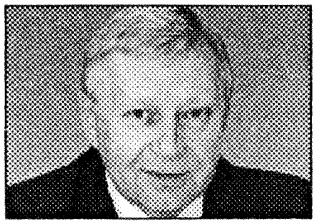
МІКРО-ПРОЦЕСОРИ ТА МІКРО-КОНТРОЛЕРИ

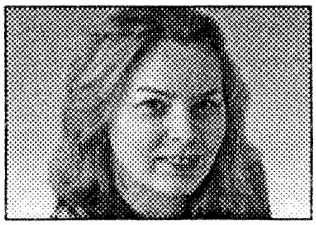
БОЙКО Віталій Іванович. Народився у 1944 р., доктор технічних наук, професор, академік Академії наук вищої школи України, заслужений діяч науки і техніки України, завідувач кафедри електроніки та автоматики Дніпродзержинського державного технічного університету. Автор понад 250 наукових і методичних праць. Фахівець у галузі електронних систем та інформаційних технологій



ГУРЖІЙ Андрій Миколайович. Народився у 1946 р., доктор технічних наук, професор, академік Академії педагогічних наук України, заслужений працівник освіти України. Автор понад 310 наукових і навчально-методичних праць. Фахівець у галузі інформаційних технологій та вимірювальної техніки

ЖУЙКОВ Валерій Якович. Народився у 1945 р., доктор технічних наук, професор, завідувач кафедри промислової електроніки Національного технічного університету України «Київський політехнічний інститут». Автор понад 270 наукових і навчально-методичних праць. Фахівець у галузі теорії та практики електронних систем

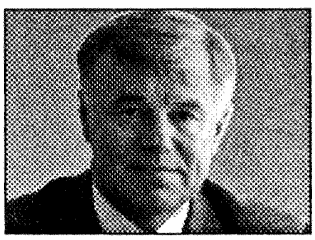
ЗОРІ Анатолій Анатолійович. Народився у 1941 р., доктор технічних наук, професор, заслужений діяч науки і техніки України, завідувач кафедри електронної техніки Донецького національного технічного університету. Автор понад 230 наукових і методичних праць. Фахівець у галузі електроніки



ПЕТЕРГЕРЯ Юлія Сергіївна. Народилася у 1975 р., кандидат технічних наук, доцент кафедри промислової електроніки Національного технічного університету України «Київський політехнічний інститут». Автор понад 60 наукових та навчально-методичних праць. Фахівець у галузі керування енергоспоживанням електротехнічних систем

СПІВАК Віктор Михайлович. Народився у 1945 р., кандидат технічних наук, професор кафедри звукотехніки та реєстрації інформації Національного технічного університету України «Київський політехнічний інститут». Автор понад 150 наукових і навчально-методичних праць. Фахівець у галузі телекомунікаційних технологій та електронних систем

ТЕРЕЩЕНКО Тетяна Олександрівна. Народилася у 1950 р., доктор технічних наук, професор кафедри промислової електроніки Національного технічного університету України «Київський політехнічний інститут». Автор понад 150 наукових і навчально-методичних праць. Фахівець у галузі теорії та практики мікропроцесорного керування електронними системами



ЯКИМЕНКО Юрій Іванович. Народився у 1945 р., доктор технічних наук, професор, член-кореспондент Національної академії наук України, заслужений діяч науки і техніки України, лауреат Державної премії у галузі науки і техніки, дійсний член ряду зарубіжних академій, почесний член американського інституту інженерів електроніки та електротехніки. Автор понад 250 наукових праць. Фахівець у галузі електроніки, радіотехніки та телекомунікацій

Книга 3

МІКРО-ПРОЦЕСОРИ ТА МІКРО-КОНТРОЛЕРИ

Затверджено Міністерством освіти і науки України

Підручник для студентів технічних спеціальностей вищих навчальних закладів

2-ге видання, доповнене і перероблене

Київ
«ВИЩА ШКОЛА»
2004

Схемотехніка електронних систем

УДК 004(075.8)
ББК 32.973я73
С92

Гриф надано Міністерством освіти і науки України (протокол від 7 квітня 2003 р. № 1/11-1366)

Автори: *В. І. Бойко, А. М. Гуржій, В. Я. Жуйков, А. А. Зорі, Ю. С. Петергеря, В. М. Співак, Т. О. Терещенко, Ю. І. Якименко*

Рецензенти: д-р техн. наук, проф. *А. А. Щерба* (Національний технічний університет України «КПІ») і канд. техн. наук, проф. *Ю. Є. Кулєшов* (Київський національний університет технологій та дизайну)

Редактор *В. С. Зацарний*

Схемотехніка електронних систем: У 3 кн. Кн. 3. Мікропроцесори та мікроконтролери: Підручник / В. І. Бойко, А. М. Гуржій, В. Я. Жуйков та ін. — 2-ге вид., допов. і переробл. — К.: Вища шк., 2004. — 399 с.: іл.

С92

ISBN 966-642-202-6 (кн. 3)

ISBN 966-642-193-3

Висвітлено загальні принципи побудови мікропроцесорів (МП), мікроконтролерів і мікропроцесорних систем. Розглянуто особливості архітектури і функціональні можливості мікропроцесорів, способи організації і побудову модулів пам'яті, принципи програмування та системи команд для різних типів МП, роботу інтерфейсів пристроїв введення-виведення інформації. Подано матеріал з архітектури, функціонального призначення і способів розширення функціональних можливостей однокристальних мікроконтролерів, керуючих контролерів і спеціалізованих мікросхем, PIC-контролерів, сигнальних процесорів та нейропроцесорів. Крім того, у другому виданні (1-ше вид. — 2003 р.) додатково наведено приклади складання програм, проектування і використання методів розширення мікропроцесорних систем.

Для студентів технічних спеціальностей вищих навчальних закладів.

УДК 004(075.8)
ББК 32.973я73

ISBN 966-642-202-6 (кн. 3)
ISBN 966-642-193-3

ЗМІСТ

ПЕРЕДМОВА

Електроніка — галузь сучасної фізики та електротехніки. Вона займається вивченням і використанням явищ, приладів і систем, основою яких є проходження електричного струму у вакуумі, газі та твердому тілі, дослідження, розробка електронних засобів і систем та принципів їх використання. Обмін інформацією в електронних системах відбувається за допомогою сигналів, носіями яких можуть бути різні фізичні величини — струми, напруги, магнітні стани, світлові хвилі. Розрізняють аналогові (безперервні) і дискретні сигнали. Є два типи дискретних сигналів: перший отримано за рівнем або за часом дискретизації безперервних сигналів, другий — у вигляді набору кодових комбінацій знаків.

Перевагами цифрових пристроїв і систем порівняно з аналоговими є підвищена завадостійкість, висока надійність, можливість тривало зберігати інформацію без її втрати, економічна й енергетична ефективність, сумісність з інтегральною технологією, висока технологічність і повторюваність, а недоліками — мала швидкодія та точність.

Основа розвитку електроніки — безперервне ускладнення функцій. На сучасному етапі стає неможливим вирішувати нові завдання старими електронними засобами з використанням існуючої елементної бази. Виникають об'єктивні умови для подальшого удосконалення елементної бази. Основними факторами є підвищення надійності, зменшення габаритних розмірів, маси, вартості та споживаної потужності.

Важливе завдання вищої освіти — правильна орієнтація майбутнього фахівця на стадії вивчення фундаментальних і професійно-орієнтованих дисциплін фаху, де поєднується як глибина важливих фізичних процесів, так і їхній розумний обсяг. Більшість випущених підручників і навчальних посібників з аналогової та цифрової схемотехніки або присвячені викладу лише окремих розділів цієї дисципліни, або дають

загальні відомості з основних розділів чи недостатньо відображають тенденції розвитку сучасної електроніки. У пропонованому підручнику автори зробили спробу ліквідації зазначених вище недоліків.

Підручник складається з трьох книг: перша книга — «Аналогова схемотехніка та імпульсні пристрої», друга — «Цифрова схемотехніка», третя — «Мікропроцесори та мікроконтролери».

Перша книга — «Аналогова схемотехніка та імпульсні пристрої» містить 12 розділів з аналогової схемотехніки та 5 розділів з імпульсних пристроїв, у яких розглянуто такі питання:

основні компоненти електронних систем, підсистем і вузлів, підсилювачі;

RC-підсилювачі напруги на біполярних і польових транзисторах за різними схемами підключення зі спільними емітером, базою, колектором, стоком, витоком;

частотні характеристики RC-підсилювачів звукових частот, робота підсилювача в області низьких, середніх та високих частот; логарифмічні амплітудно-частотні характеристики, приклади розрахунків;

узгодження джерела сигналу з навантаженням, класифікація одно- і двотактних підсилювачів потужності та підсилювачів без трансформаторів;

наскрізні характеристики каскадів, вплив температури на характеристики біполярних транзисторів, причини та методи розрахунку нелінійних спотворень;

класифікація паралельних і послідовних зворотних зв'язків: за струмом і напругою, жорстких і гнучких, їхній вплив на схемні функції, показники роботи, умови стійкості системи;

підсилення постійного струму, способи зменшення дрейфу нуля, підсилювачі на несівній частоті, з безпосередніми зв'язками, паралельні балансові та диференціальні схеми;

класифікація аналогових мікроелектронних структур, операційні підсилювачі на інтегральних мікросхемах, елементи їх схемотехніки;

побудова вирішальних структур на базі операційних підсилювачів, лінійні та нелінійні функціональні перетворювачі, суматори, інтегратори, диференціатори, частотна корекція, логарифмування, помножувачі, подільники, випрямлячі, детектори;

загальні положення теорії селективних підсилювачів різних типів;

LC-генератори періодичних коливань на польових і біполярних транзисторах;

основи теорії RC-генераторів з різними типами фазообертачів і без них;

проходження імпульсів через ланки інтегрування, диференціювання, розділові; фіксатори рівня;

формувачі прямокутних імпульсів, ключі, обмежувачі, моделі великого сигналу;

мульти- та одновібратори; регулювання частоти, термостабілізація і поліпшення форми вихідної напруги схем; генератори лінійно змінюваної напруги, блокінг-генератори в автоколивальному режимі та режимі очікування;

аналіз кодувальних пристроїв, АЦП та ЦАП, пристрої вибірки збереження;

імпульсні джерела живлення, елементна база силової електроніки та перспективи розвитку.

Друга книга — «Цифрова схемотехніка» — охоплює 12 розділів, у яких розглянуто такі питання:

математичні основи цифрової схемотехніки, системи числення, коди, двійкова арифметика та форми подання чисел;

теоретичні основи синтезу цифрових автоматів та алгебра логіки;

аналіз методів мінімізації булевих функцій, методи Карно — Вейча, Квайна, Мак-Класкі;

класифікація логічних елементів цифрових пристроїв (базові логічні елементи);

синтез комбінаційних схем, мульти- та демультиплексори, суматори, шифратори, дешифратори, компаратори, перетворювачі кодів;

асинхронні, синхронні тригерні елементи, RS-, D-, JR-тригери;

синтез цифрових автоматів, регістри зсуву, лічильники, цифрові фазообертачі;

логічні розширювачі, перетворювачі рівнів, таймери;

статичні, динамічні оперативні та мікросхеми постійних запам'ятовувальних пристроїв;

проектування логічних схем, перехідні процеси, гонки, одно- та двофазова синхронізація;

застосування цифрових інтегральних мікросхем, завади і завадостійкість, монтаж цифрових інтегральних мікросхем.

Третя книга — «Мікропроцесори та мікроконтролери» — складається з 9 розділів, у яких розглянуто такі питання:

загальні принципи побудови мікропроцесорних систем, організація шин, поняття про архітектуру мікропроцесорів та основні принципи побудови мікропроцесорних систем, основи програмування мовою асемблер;

однокристальні 8- та 16-розрядні мікропроцесори, відомості про систему команд мікропроцесора *i*8086;

старші моделі однокристальних універсальних мікропроцесорів (*i*80286, *i*386, *i*486, архітектура мікропроцесорів *Pentium*);

системи пам'яті: класифікація постійних та оперативних запам'ятовувальних пристроїв, побудова модулів пам'яті, принципи організації стекової та кеш-пам'яті;

інтерфейс пристроїв введення-виведення — паралельний та послідовний інтерфейс, контролер клавіатури та індикації, програмовний таймер, контролер прямого доступу до пам'яті, контролер переривань;

архітектура, функціональні можливості та система команд однокристальних мікроконтролерів з CISC-архітектурою, розширення можливостей, приклад застосування для керування двигуном постійного струму;

однокристальні мікроконтролери з RISC-архітектурою: PIC-контролери, AVR-мікроконтролери;

сигнальні мікропроцесори оброблення даних у форматі з фіксованою та плаваючою комою, їхні характеристики і функціональні можливості;

нейронні обчислювачі та їхні функції, основи побудови нейронних мереж, алгоритми навчання, апаратна реалізація.

У підручнику в стислому вигляді та доступній формі викладено всі розділи програми підготовки бакалаврів, інженерів та магістрів напряму «Електроніка» спеціальності «Електронні системи» і «Фізична та біомедична електроніка» згідно з вимогами державного стандарту України. Це може підвищити ефективність не лише аудиторних занять, а й самостійної роботи студентів. Матеріал скомпонований так, що кожний наступний розділ є логічним продовженням попереднього.

У результаті вивчення курсу студенти засвоюють принципи функціонування, вибору, практичної реалізації пристроїв та систем електроніки різного призначення, методи їх аналізу і розрахунку за заданими статичними й динамічними параметрами та принципи розроблення систем керування електронними системами. Студент має знати: принципи побудови та функціонування пристроїв аналогової і цифрової схемотехніки; принципи вибору методів аналізу і розрахунку електронних пристроїв із заданими характеристиками; принципи побудови і функціонування мікропроцесорних та мікроконтролерних систем, а також уміти: розрахувати електронні ланки; узагальнити динамічні показники електронних пристроїв; виконати розрахунки різних електронних пристроїв з організацією банку даних, розробити структурні та принципові схеми, а також програмне забезпечення

мікропроцесорних систем керування пристроями електроніки.

Підручник написано на основі досвіду викладання дисциплін згідно з програмами підготовки бакалаврів, інженерів та магістрів напряму «Електроніка» в Національному технічному університеті України «КПІ», Донецькому національному технічному університеті та Дніпродзержинському державному технічному університеті.

Курс забезпечується основними дисциплінами: математика, фізика, теоретичні основи електротехніки, твердотіла електроніка.

Автори висловлюють вдячність співробітникам кафедр «Промислова електроніка» Київського НТУУ «КПІ», «Електронна техніка» Донецького НТУ і «Електроніка та автоматика» Дніпродзержинського ДТУ за допомогу під час підготовки оригіналу-макета та обговорення навчального матеріалу.

Автори щиро вдячні рецензентам за цінні зауваження та рекомендації щодо вдосконалення окремих розділів рукопису, які вони врахували під час його доопрацювання, що сприяло поліпшенню змісту підручника.

ВСТУП

Важливе місце у схемотехніці електронних систем посідають системи керування з мікропроцесорами і мікроконтролерами, які дають змогу реалізувати складні закони керування електронними пристроями. Знання схемотехніки аналогових і цифрових систем створює базу для вивчення принципів побудови мікропроцесорних систем керування. Перевагою мікропроцесорних систем керування є їх гнучкість, тобто систему, розроблену для розв'язання певної задачі керування, легко пристосувати для розв'язання інших задач, змінивши програмне забезпечення.

Перший мікропроцесор Intel 4004 було створено у 1971 р. Він працював на частоті 750 кГц, мав 2300 транзисторів і чотирирозрядну шину даних. Цей винахід визнано одним з найбільших досягнень XX ст. Сучасні мікропроцесори — великі інтегральні схеми (ВІС) або однокристальні мікроконтролери — містять усі складові ЕОМ (мікропроцесори, пам'ять даних та пам'ять програм, інтерфейсні схеми) і їх ефективно використовують у системах керування промислового та побутового обладнання.

Розширення функцій мікропроцесорних систем потребувало вдосконалення знань спеціалістів різноманітних профілів у цьому напрямі. Тому вивчення основ побудови і програмування мікропроцесорів — необхідна складова підготовки спеціалістів у вищих навчальних закладах. Незважаючи на велику різноманітність типів мікропроцесорів та функцій, що виконуються ними, логіка побудови систем і створення програмного забезпечення залишається незмінною. Вивчення загальних принципів побудови, особливостей архітектури, використання різних видів пам'яті та програмування мікропроцесорних комплектів дає теоретичну базу для розробки і використання мікропроцесорних систем.

Мета підручника — набуття студентами теоретичних знань і практичних навичок побудови та програмування мікропроцесорних систем керування, проектування інтерфейсів введення-виведення, спеціалізованих пристроїв на мікропроцесорах різних серій та мікро-ЕОМ. Підручник складено за матеріалами лекцій з дисциплін, пов'язаних з мікропроцесорною технікою та її використанням і включених у навчальний процес Національного технічного університету України «Київський політехнічний інститут».

У підручнику наведено загальні принципи побудови МП, мікроконтролерів та мікропроцесорних систем; розглянуто особливості архітектури і функціональні можливості 8-розрядних (*i*8080, *i*8085), 16-розрядних (*i*8086/88, *i*80186/188, *i*80286), 32-розрядних мікропроцесорів (*i*386, *i*486, Pentium) та 64-розрядних процесорів (IA-64, McKinley, Itanium); способи організації і побудову модулів пам'яті; принципи програмування і системи команд мови асемблер для різних типів МП; роботу інтерфейсів пристроїв введення-виведення (ПВВ) інформації. Подано матеріал з архітектури, функціонального призначення і способів розширення функціональних можливостей однокристальних мікроконтролерів (ОМК), керуючих контролерів та спеціалізованих мікросхем, а також PIC-контролерів, сигнальних процесорів і нейропроцесорів. З метою кращого сприйняття матеріалу в підручнику наведено приклади складання програм, проектування мікропроцесорних систем, використання методів розширення можливостей системи, а після кожного підрозділу дано контрольні запитання.

Автори висловлюють подяку за допомогу в підготовці та оформленні матеріалу підручника студентам-магістрам кафедри промислової електроніки НТУУ «КПІ» Т. В. Хижняк і Д. Д. Кункіну.

Розділ 1

ЗАГАЛЬНІ ПРИНЦИПИ ПОБУДОВИ МІКРОПРОЦЕСОРНИХ СИСТЕМ

1.1. Основні поняття і визначення

Мікропроцесор (*МП*) — пристрій, який здійснює приймання, оброблення і видачу інформації. Конструктивно МП містить одну або кілька інтегральних схем і виконує дії за програмою, записаною в пам'яті.

Мікропроцесорна система — обчислювальна, контрольно-вимірювальна або керуюча система, в якій основним пристроєм обробки інформації є МП. Мікропроцесорна система складається з набору великих інтегрованих схем (ВІС).

Мультимікропроцесорна, або *мультипроцесорна, система* — система, яка утворюється об'єднанням деякої кількості універсальних або спеціалізованих МП, завдяки чому забезпечується паралельна обробка інформації і роздільне керування.

Мікропроцесорний комплект (*МПК*) — сукупність інтегральних схем, сумісних за електричними, інформаційними та конструктивними параметрами і призначених для побудови електронно-обчислювальної апаратури й мікропроцесорних систем керування. Типовий склад МПК: ВІС МП (один чи кілька корпусів інтегральних схем); ВІС оперативних запам'ятовувальних пристроїв (ОЗП); ВІС постійних запам'ятовувальних пристроїв (ПЗП); інтерфейси, або контролери, зовнішніх пристроїв; службові ВІС (тактовий генератор, регістри, шинні формувачі, контролери та арбітри шин).

Мікропроцесори та мікропроцесорні комплекти класифікують за такими ознаками: призначенням, кількістю ВІС, способом керування, типом архітектури та типом системи подавання команд.

За призначенням МП поділяють на універсальні та спеціалізовані. *Універсальними мікропроцесорами* є мікропроцесори загального призначення, які дають змогу розв'язувати широкий клас задач — обчислення, оброблення та керування. *Спеціалізовані мікропроцесори* призначені для розв'я-

зання задач лише певного класу. До них належать сигнальні, медійні та мультимедійні МП; трансп'ютери.

Сигнальні процесори призначені для цифрового оброблення сигналів у реальному масштабі часу (наприклад, фільтрація сигналів, обчислення згортки, обчислення кореляційної функції, підсилення, обмеження і трансформація сигналу, пряме і зворотне перетворення інтегралів Фур'є). До сигнальних процесорів належать процесори фірм Texas Instruments — TMS320C80, Analog Devices — ADSP2106x, Motorola — DSP560xx та DSP9600x.

Медійні й мультимедійні процесори використовують для оброблення аудіосигналів, графічної інформації, відеозображень та для розв'язання задач у мультимедіакомп'ютерах, іграшкових приставках, побутовій техніці. До медійних і мультимедійних процесорів належать процесори фірм MicroUnity — Mediaprocessor, Philips — Trimedia, Cromatic Reserch — Mpact Media Engine, Nvidia — NV1, Cyrix — MediaGX.

Трансп'ютери застосовують для масових паралельних обчислень і роботи у мультипроцесорних системах. Для них характерним є наявність внутрішньої пам'яті та вбудованого міжпроцесорного інтерфейсу, тобто каналів зв'язку з іншими ВІС МП. До трансп'ютерів належать процесори фірм Inmos — T-2, T-4, T-8, T9000.

За кількістю ВІС у МПК розрізняють багатокристальні МПК та однокристальні мікроконтролери (ОМК). Багатокристальні комплекти — це МПК з однокристальними і секційними МП.

Однокристальний мікропроцесор, або *мікропроцесор з фіксованою розрядністю даних*, є конструктивно завершеним виробом у вигляді однієї ВІС. До цього типу належать процесори фірм Intel — Pentium (P5, P6, P7), AMD — K5, K6,Cyrix — 6x86,Diqital Equipment — Alpha 21064, 21164A, Silicon Graphics — MIPS R10000, Motorola — Power PC 603, 604, 620, Hewlett Packard — PA-8000, Sun Microsystems — Ultra SPARC II.

У *секційних мікропроцесорах* в одній ВІС реалізується лише деяка функціональна частина (секція) процесора. Інша назва секційних МП — *розрядно-модульні мікропроцесори*, або *мікропроцесори з нарощенням розрядності*. Секційність ВІС МП зумовлює значну гнучкість МПС, можливість нарощення розрядності даних, створення специфічних технологічних команд із набору мікрокоманд. До секційних належать МП серій К589, К1804.

Однокристальний мікроконтролер — пристрій, що конструктивно виконаний в одному корпусі ВІС і містить основ-

ні складові МПК. До таких мікроконтролерів належать ОМК фірм Intel — MCS-196/296, MicroChip — PIC17C4x, PIC17C75x, Mitsubishi Electric — M3820, Motorola — MC33035, MC33039.

За способом керування розрізняють МП зі схемним та МП з мікропрограмним керуванням. *Мікропроцесори зі схемним керуванням* мають фіксований набір команд, розроблений фірмою-виробником, який не може змінюватися споживачем. У *мікропроцесорах з мікропрограмним керуванням* систему команд розробляють під час проектування конкретного МПК на базі набору найпростіших мікрокоманд з урахуванням класу задач, для розв'язання яких призначений МПК.

За типом архітектури, або принципом побудови, розрізняють МП з фоннеймановською архітектурою і МП з гарвардською архітектурою.

За типом системи команд розрізняють *CISC* (*Complete Instruction Set Computing*) — процесори з повним набором команд і *RISC* (*Reduced Instruction Set Computing*) — процесори зі зменшеним набором команд.

Слід зазначити, що багато МПК підпадають під різні класифікаційні ознаки, оскільки здатні вирішувати задачі різних класів. Так, існують універсальні МП з мультимедійним розширенням наборів команд, наприклад, Pentium MMX, Pentium II, Cyrix 6x86MX, AMD K6, Ultra SPARC. У CISC-процесорах Pentium PRO реалізовано ядро з RISC-архітектурою.

1.2. Організація шин

Шина — інформаційний канал, що об'єднує всі функціональні блоки МПС і забезпечує обмін даними у вигляді двійкових чисел. Конструктивно шина складається з *n* провідників та з одного спільного провідника (землі). Дані передаються по шині у вигляді слів, що є групою бітів.

У *паралельній шині n* бітів передаються по окремих лініях одночасно, у *послідовній* — по єдиній лінії послідовно у часі. Паралельні шини виконують у вигляді плоского кабелю, а послідовні — у вигляді коаксіального або волоконно-оптичного кабелю. Коаксіальний кабель використовують для передавання даних на відстань до 100 м, узгоджуючи передавальні та приймальні каскади з хвильовим опором лінії. Волоконно-оптичний кабель використовують для передавання на значно більші відстані.

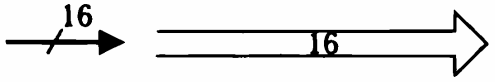


Рис. 1.1. Варіанти умовних позначень однонапрямленої паралельної 16-розрядної шини

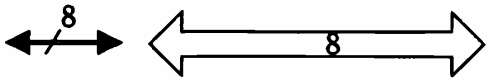


Рис. 1.2. Варіанти умовних позначень двонапрямленої паралельної 8-розрядної шини

Усі основні блоки МПС з'єднані з єдиною паралельною шиною, яку називають *системною шиною SB* (*System Bus*). Системна шина містить три шини — адреси, даних і керування.

Шина адреси AB (*Address Bus*) є однонапрямленою. Вона призначена для передавання адреси комірки пам'яті або пристрою введення-виведення. Напрям передавання по шині адреси — від МП до зовнішніх пристроїв. Варіанти умовних позначень однонапрямленої паралельної шини показано на рис. 1.1, на якому стрілка вказує напрям передавання.

Число 16 на рис. 1.1 означає розрядність шини. Зазначимо, що допускається позначення шин і без наведення розрядності.

Шина даних DB (*Data Bus*) є двонапрямленою. Вона призначена для передавання даних між блоками МПС. Інформація по одних і тих самих лініях *DB* може передаватися у двох напрямах — як до МП, так і від нього. Варіанти умовних позначень двонапрямленої шини показано на рис. 1.2.

Шина керування CB (*Control Bus*) призначена для передавання керуючих сигналів. Хоча напрям керуючих сигналів може бути різним, однак шина керування не є двонапрямленою, оскільки для сигналів різного напряму використовуються окремі лінії.

1.3. Принципи побудови мікропроцесорних систем

В основу побудови мікропроцесорних систем (МПС) покладено три принципи — магістральності, модульності, мікропрограмного керування.

Принцип магістральності визначає характер зв'язків між функціональними блоками МПС — усі блоки з'єднані з єдиною системною шиною.

Принцип модульності полягає в тому, що система будується на основі обмеженої кількості типів конструктивно і функціонально завершених модулів. Кожний модуль МП системи має вхід керування третім (високоімпедансним) станом. Цей вхід називається $\overline{CS}$ (*Chip Select*) — вибір кристала або $\overline{OE}$ (*Output Enable*) — дозвіл виходу.

Дію сигналу $\overline{CS}$ для тригера показано на рис. 1.3. Вихідний сигнал тригера Q з'являється на виводі лише за активного (у цьому випадку — нульового) рівня сигналу $\overline{CS}$. Якщо $\overline{CS} = 1$, тригер переводиться у високоімпедансний стан. Вихід тригера є *тристабільним*, тобто може перебувати в одному з трьох станів: логічної одиниці, логічного нуля або у високоімпедансному. У кожний момент часу до системної шини МПС під'єднано лише два модулі — той, що приймає, і той, що передає інформацію. Інші знаходяться у високоімпедансному стані.

Принципи магістральності та модульності дають змогу нарощувати керуючі та обчислювальні можливості МП через під'єднання інших модулів.

Принцип мікропрограмного керування полягає у можливості здійснення елементарних операцій — мікрокоманд (зсуву, пересилання інформації, логічних операцій). Певною комбінацією мікрокоманд можна створити набір команд, який максимально відповідатиме призначенню системи, тобто створити *технологічну мову*. У секційних процесорах набір мікрокоманд можна змінити, використовуючи інші мікросхеми пам'яті мікрокоманд.

Узагальнену структурну схему МПС зображено на рис. 1.4. До складу МПС входять центральний процесор (ЦП), ПЗП, ОЗП, система переривань, таймер, ПВВ. Пристрої введення-виведення під'єднані до системної шини через інтерфейси введення-виведення.

Постійний та оперативний запам'ятовувальні пристрої — це *система пам'яті*, яку використовують для збереження інформації у вигляді двійкових чисел. Постійний запам'ятовувальний пристрій призначений для збереження програм керування таблиць і констант, а оперативний запам'ятовувальний пристрій (ОЗП) — для збереження проміжних результатів обчислень. Пам'ять організовано у вигляді масиву комірок, кожна з яких має свою адресу і містить байт або слово. *Байтом* називають групу із 8 біт. Слово може мати будь-яку довжину в бітах. Під *словом* найчастіше розуміють двійкове число завдовжки два байти.

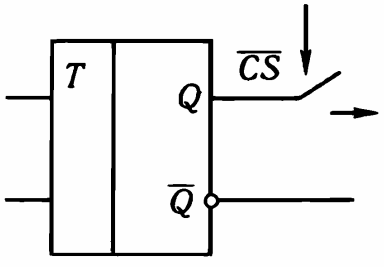


Рис. 1.3. Дія сигналу $\overline{CS}$ для тригера

Для звернення до комірки пам'яті треба видати її адресу на шину адреси. На рис. 1.5 зображено структуру пам'яті з 8 однобайтових комірок, де кожній адресі відповідає певний вміст комірки. Так, комірка з адресою 000 має вміст $01011111_2 = 5F_{16}$.

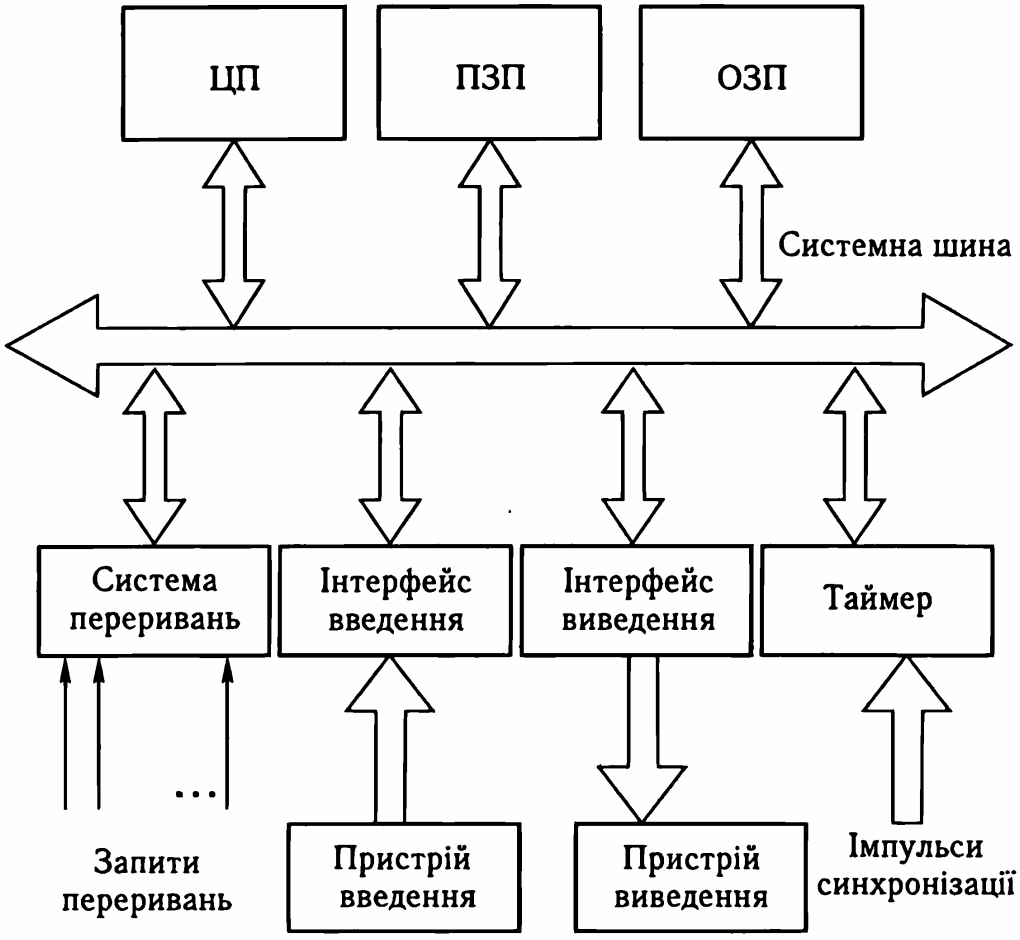


Рис. 1.4. Узагальнена структурна схема мікропроцесорної системи керування

Сегментом називають область пам'яті, яка починається з будь-якої адреси, кратної 16, і займає до 64 Кбайт. Існують три основних сегменти: сегмент кодів, сегмент даних, сегмент стеку.

Сегмент кодів містить коди команд, які адресуються сегментним регістром кодів *CS* та регістром — лічильником команд *IP*. Регістр *CS* визначає початкову адресу сегмента кодів, а регістр *IP* — зміщення в сегменті (відстань від початку сегмента до комірки, в якій знаходиться адреса команди).

Сегмент даних містить дані, константи та робочі області, необхідні для виконання програми. Регістр *DS* має початкову адресу сегмента даних, а зміщення в сегменті задається в команді.

Сегмент стеку містить адреси повернення з підпрограм та дані. Регістр *SS* має початкову адресу сегмента стеку, а регістр *SP* — зміщення в сегменті.

У деяких операціях використовують *додатковий сегмент даних*, початкова адреса якого задається регістром *ES*, а зміщення в сегменті — командою.

Адреса	Дані
000	01011111
001	00010011
010	01110111
011	00001100
100	00000000
101	11111111
110	10101010
111	11110000

Рис. 1.5. Структура пам'яті з 8 однобайтових комірок

Двобайтове зміщення (16 біт) може знаходитися в межах від 0000_{16} до $FFFF_{16}$. Для звернення до будь-якої адреси у програмі виконується додавання адрес, які знаходяться в регістрі сегмента та зміщення. Наприклад, перший байт у сегменті кодів має зміщення нуль, другий байт — одиницю і так аж до $FFFF_{16}$.

Конкретна адреса команди (для сегмента кодів), комірки пам'яті (для сегмента даних та додаткового сегмента) або комірки стеку (для сегмента стеку) визначається результатом додавання адреси сегмента, яка міститься у відповідному сегментному регістрі, та зміщення.

Модуль центрального процесора здійснює оброблення даних і керує всіма іншими модулями системи. Крім ВІС МП, центральний процесор містить схеми синхронізації та інтерфейсу із системною шиною. Він вибирає коди команд з пам'яті, дешифрує їх і виконує. Впродовж часу виконання команди — *командного циклу* — центральний процесор (ЦП) виконує такі дії:

- виставляє адресу команди на шину адреси *AB;*
- отримує код команди з пам'яті та дешифрує його;
- обчислює адреси операнда і зчитує дані;
- виконує операцію, визначену командою;
- сприймає зовнішні керуючі сигнали, наприклад запити переривань;
- генерує сигнали стану і керування, необхідні для роботи пам'яті та пристрою введення-виведення (ПВВ).

Пристрої введення-виведення, або *зовнішні пристрої*, — це пристрої, призначені для введення інформації в МП або виведення інформації з нього. Прикладами ПВВ є дисплеї, друкувальні пристрої, клавіатура, цифроаналоговий та аналого-цифровий пристрої, реле, комутатори. Для з'єднання ПВВ із системною шиною їхні сигнали мають відповідати певним стандартам. Це досягається за допомогою інтерфейсів введення-виведення.

Інтерфейси введення-виведення називають також *контролерами,* або *адаптерами.* Мікропроцесор звертається до інтерфейсів за допомогою спеціальних команд введення-виведен-

ня. При цьому МП виставляє на шину адреси *AB* адресу інтерфейсу і по шині даних *DB* зчитує дані з пристрою введення або записує у пристрій виведення. На рис. 1.4 показано один інтерфейс введення та один інтерфейс виведення.

Система переривань дає змогу МПС реагувати на зовнішні сигнали — запити переривань, джерелами яких можуть бути сигнали готовності від зовнішніх пристроїв, сигнали від генераторів та сигнали з виходів датчиків. З появою запиту переривання ЦП переривае основну програму і переходить до виконання підпрограми обслуговування запиту переривання. Для побудови системи переривань МПК містять ВІС спеціальних програмованих контролерів переривань.

Таймер призначений для реалізації функцій, пов'язаних з відліком часу. Після того як МП завантажує в таймер число, що задає частоту, затримку або коефіцієнт ділення, таймер реалізує необхідну функцію.

1.4. Архітектура мікропроцесорів

Поняття *архітектури мікропроцесора* визначає його складові частини та зв'язки і взаємодію між ними. Архітектура містить: структурну схему самого МП; програмну модель МП (описання функцій регістрів); інформацію про організацію пам'яті (ємність пам'яті та способи її адресації); опис організації процедур введення-виведення.

Існують два основних типи архітектури — фоннеймановська і гарвардська. *Фоннеймановську архітектуру* (рис. 1.6, *а*) у 1945 р. запропонував американський математик Джо фон Нейман. Особливістю цієї архітектури є те, що програма і дані знаходяться у спільній пам'яті, доступ до якої здійснюється по одній шині даних і команд.

Гарвардську архітектуру вперше запроваджено у 1944 р. в релейній обчислювальній машині Гарвардського університету (США). Особливістю цієї архітектури є те, що пам'ять даних і пам'ять програм розділені і мають окремі шину даних і шину команд (рис. 1.6, *б*), що дає змогу підвищити швидкодію МП системи.

Структурні схеми обох архітектур містять: процесорний елемент, пам'ять, інтерфейси введення-виведення (ІВВ) і ПВВ. Пам'ять і ІВВ для різних типів МП можуть бути як внутрішніми, тобто розміщуватися на тому самому кристалі, що й процесорний елемент, так і зовнішніми. Процесорний елемент містить регістри, арифметико-логічний пристрій (АЛП), пристрій керування і виконує функції оброблення даних та

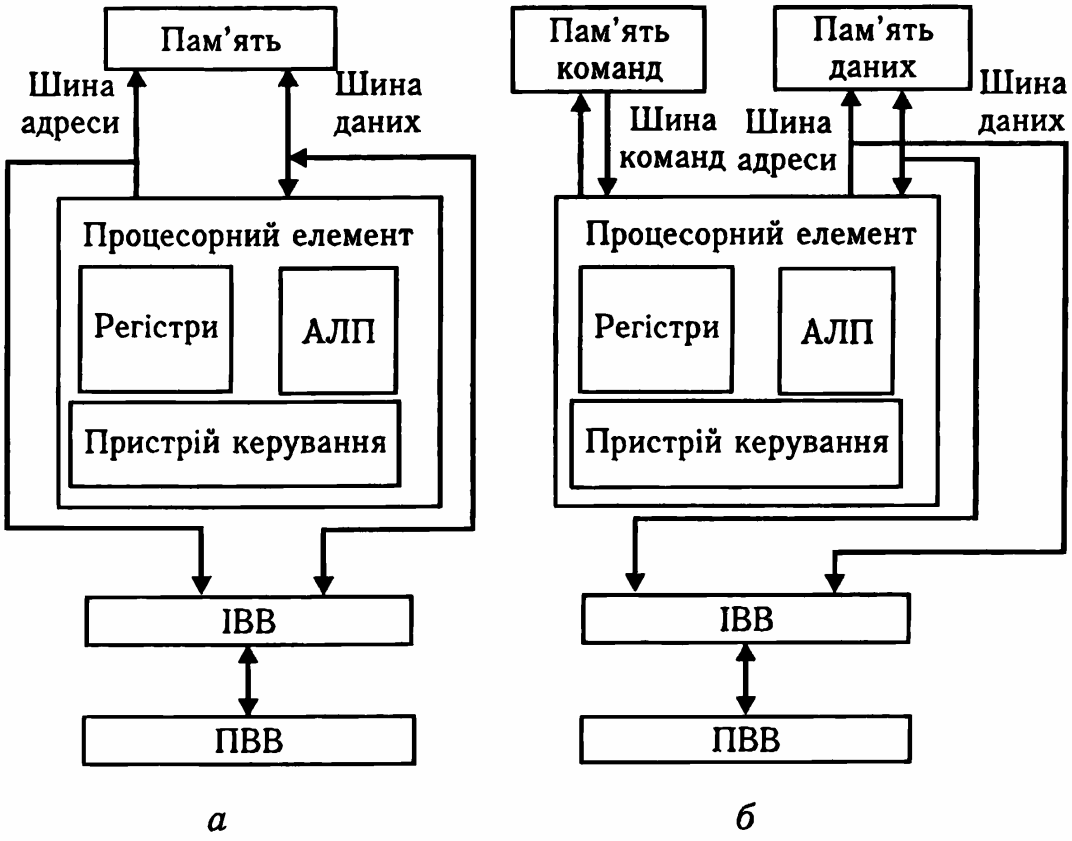


Рис. 1.6. Основні типи архітектури:
а — фоннейманівська; *б* — гарвардська

керування процесами обміну інформацією. Пам'ять забезпечує зберігання кодів команд програми і даних. Інтерфейси призначені для зв'язку з ПВВ, наприклад з клавіатурою, дисплеєм, друкувальними пристроями, датчиками інформації. Усі елементи структурної схеми з'єднані за допомогою шин.

Розширену структурну схему з процесором фоннейманівської архітектури зображено на рис. 1.7.

Схема процесора містить пристрій керування, АЛП і регістри: адреси, даних, команд, а також стану, акумулятор, лічильник команд, покажчик стеку.

Пристрій керування відповідно до кодів команд і зовнішніх керуючих сигналів та сигналів синхронізації формує керуючі сигнали для всіх блоків структурної схеми МП, а також керує обміном інформацією між МП, пам'яттю і ПВВ. Пристрій керування реалізує такі функції: початкового встановлення МП, синхронізації, переривань, узгодження швидкодії модулів МП системи.

Функція початкового встановлення МП. Зовнішній сигнал початкового встановлення процесора *RESET* формується під час ввімкнення джерела живлення МП або під час натискання кнопки *RESET*. Після появи цього сигналу пристрій ке-

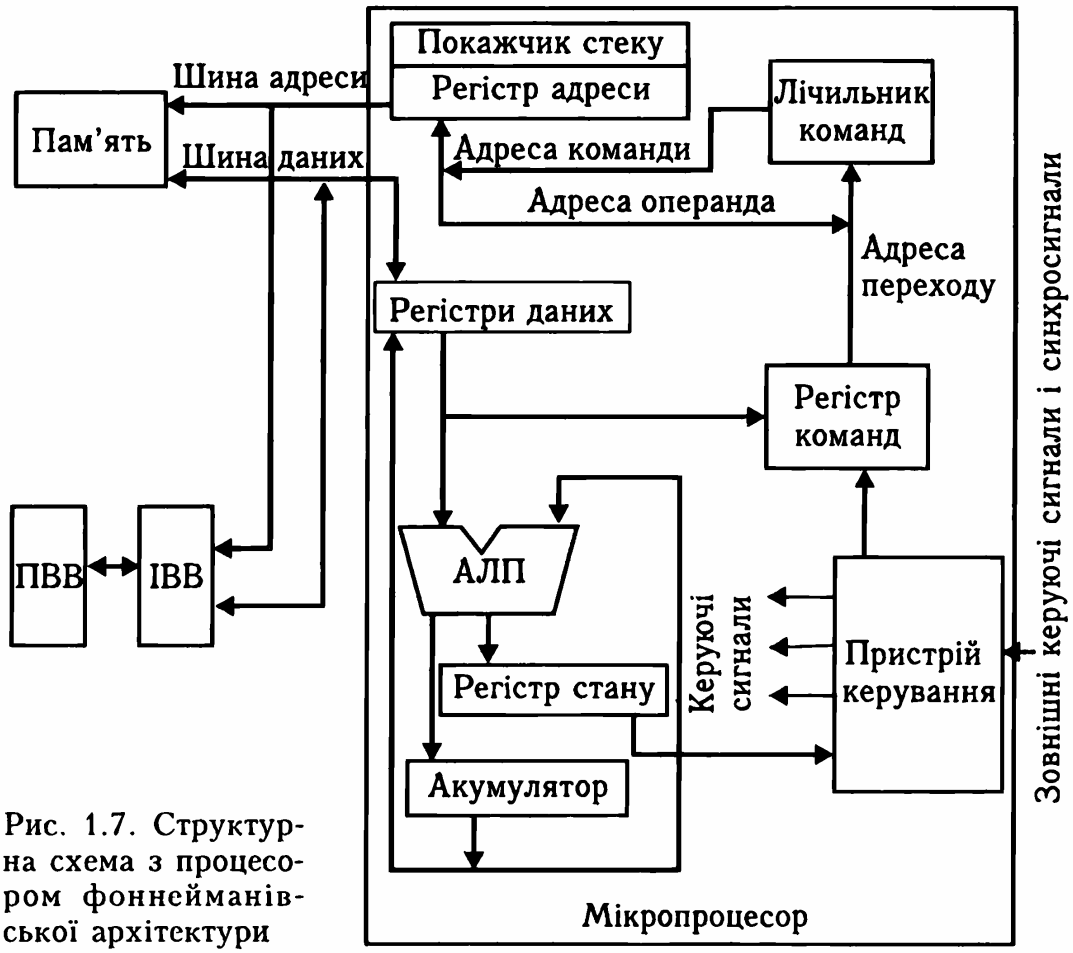


Рис. 1.7. Структурна схема з процесором фоннейманівської архітектури

рування забезпечує завантаження нульового значення у програмний лічильник, що ініціює вибирання з пам'яті байта команди з нульовою адресою. Наприкінці вибирання вміст лічильника команд збільшується на одиницю, після чого вибирається байт команд з наступною адресою. Так виконується вся записана у пам'яті програма.

Функція синхронізації. Згідно із зовнішніми керуючими сигналами і сигналами синхросигналізації пристрій керування синхронізує роботу всіх блоків МП.

Функція переривань. З надходженням сигналу переривання пристрій керування ініціює роботу підпрограми обробки відповідного переривання. Потреба у реалізації функцій переривань виникає тоді, коли під час виконання основної програми треба перевести МП на розв'язання іншої задачі, наприклад оброблення аварійної ситуації або роботи з ПВВ.

Функція узгодження швидкодії модулів мікропроцесорної системи. Під час обслуговування пам'яті та ПВВ із значно меншою швидкодією, ніж МП, узгодження швидкодії вирішується генерацією тактів очікування МП, а під час обслуговування пристроїв з більшою швидкодією, ніж МП, використовується режим безпосереднього доступу до пам'яті.

Арифметико-логічний пристрій (АЛП) — комбінаційна схема на основі суматора, який сигналами з виходів пристрою керування налагоджується на виконання певної арифметичної або логічної операції — додавання, віднімання, ЛОГІЧНЕ І, ЛОГІЧНЕ АБО, логічне НІ, ВИКЛЮЧАЛЬНЕ АБО, зсуву, порівняння, десяткової корекції. Отже, АЛП виконує арифметичні або логічні операції над операндами, які пересилаються з пам'яті і(або) регістрів МП.

Операнд — це об'єкт у вигляді значення даних, вмісту регістрів або вмісту комірки пам'яті, з яким оперує команда, наприклад, у команді додавання операндами є доданки. Операнд може задаватися у команді у вигляді числа або знаходитися в регістрі чи комірці пам'яті. Отриманий після виконання команди в АЛП результат пересилається в регістр або комірку пам'яті.

Регістри призначені для зберігання n-розрядного двійкового числа. Це n тригерів зі схемами керування читанням-записом та вибірки. Регістри створюють внутрішню пам'ять МП і використовуються для зберігання проміжних результатів обчислень.

Акумулятор — регістр, в якому зберігається один з операндів. Після виконання команди в акумуляторі замість операнда розміщується результат операції. У 8-розрядних процесорах акумулятор бере участь у всіх операціях АЛП пристрою. Однак у 16-розрядних МП більшість команд виконуються без участі акумулятора, але в деяких командах (введення, виведення, множення, ділення) акумулятор діє так само, як і в 8-розрядних МП, тобто зберігає один з операндів, а після виконання команди — результат операції.

Лічильник команд, або ***програмований лічильник,*** призначений для зберігання адреси комірки пам'яті, яка містить код наступної команди. Програма дій МП записана в пам'яті у вигляді послідовності кодів команд. Для переходу до наступної команди вміст лічильника збільшується на одиницю у момент вибирання команди з пам'яті. Отже, наприкінці виконання команди в лічильнику команд зберігається адреса наступної команди.

Покажчик стеку — це регістр, який зберігає адресу останньої зайнятої комірки стеку. *Стеком,* або *стековою пам'яттю,* називають область пам'яті, яка організована за принципом «останній прийшов — перший пішов».

Регістр команд зберігає код команди впродовж усього часу виконання команди.

Регістр адреси і ***регістри даних*** призначені для зберігання адрес та даних, що використовуються під час виконання поточної команди в МП.

Регістр стану, або ***регістр прапорців*** (ознак), призначений для зберігання інформації про результат операції в АЛП і складається з кількох тригерів, які набувають одиничних або нульових значень. Наприклад, прапорець нуля встановлюється у стан логічної одиниці за нульового результату операції.

1.5. Основи програмування мовою Асемблер

Програма — послідовність команд, виконання яких приводить до розв'язання задачі.

Команда визначає операцію, яку має виконати МП з даними. Команда містить у явній або неявній формах інформацію про те, де буде розташований результат операції, та про адресу наступної команди. Код команди складається з кількох частин, які називають *полями*. Склад, призначення і розташування полів називається *форматом команди*. У загальному випадку формат команди містить операційну та адресну частини. Операційна частина містить код операції, наприклад додавання, множення, передавання даних. Адресна частина складається з кількох полів і має інформацію про адреси операндів, результати операції та наступні команди. Формат команди, в якому адресна частина складається з двох полів (ознаки адресації та адреси операндів), показано на рис. 1.8.

Поле «ознака адресації» визначає спосіб адресації операнда. Біти полів «Ознака адресації» та «Адреси операндів» разом визначають комірки пам'яті, в яких зберігаються операнди.

Розрізняють такі групи команд: 1) команди передавання даних; 2) команди операцій введення-виведення; 3) команди оброблення інформації (арифметичні, логічні, зсуву, порівняння операндів, десяткової корекції); 4) команди керування порядком виконання програми (переходу, викликів підпрограм, повернення з підпрограм, переривань); 5) команди задання режимів роботи МП.

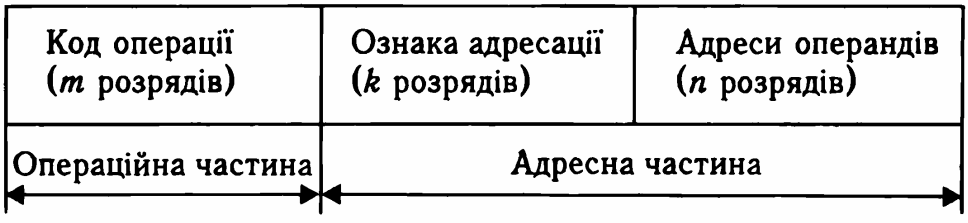


Рис. 1.8. Формат команди

Загальну кількість бітів у коді команди називають *довжиною формату*. Кількість двійкових розрядів m у полі «код операції» забезпечує можливість подання всіх операцій, які виконує МП. Якщо МП виконує M різних операцій, то кількість розрядів визначається так:

$$m \geq \log_2 M.$$

Якщо пам'ять містить S комірок, то потрібна для запису адреси одного операнда кількість розрядів у полі «адреси операндів» становить:

$$n \geq \log_2 S.$$

Довжина формату команди визначає швидкість виконання команди і залежить від способу адресації операндів. Існують такі способи адресації: пряма, непряма, безпосередня, автоінкрементна (автодекрементна), сторінкова, індексна, відносна.

Пряма адресація. За такої адресації адреса операнда безпосередньо вказана в команді. Як приклад розглянемо команду МП К580ВМ80А (*i*8080) прямого завантаження акумулятора вмістом комірки пам'яті, розміщеної за адресою 0012_{16}. Формат і схему виконання цієї команди показано на рис. 1.9.

У байті 1 команди (рис. 1.9, *а*) міститься код операції пересилання даних в акумулятор з комірки пам'яті, а в байтах 2 і 3 — адреса комірки пам'яті. У байті 2 розміщений молодший (12_{16}), а в байті 3 — старший (00_{16}) байти адреси.

На рис. 1.9, *б* комірка пам'яті з адресою 0012_{16} має вміст 11010111_2. Вміст акумулятора до операції становить 00000000_2. Після виконання команди значення вмісту комірки пам'яті копіюється в акумуляторі.

Непряма адресація. За такої адресації у форматі команди вказується номер регістра, в якому зберігається адреса комірки пам'яті, що містить операнд. Для збереження 16-розрядної адреси у 8-розрядному процесорі 8-розрядні регістри об'єднані у *регістрові пари*. У першому регістрі регістрової пари зберігається старший байт адреси, а в другому — молодший. Номер регістрової пари, в якій зберігається адреса, є дворозрядним двійковим числом, тому він розміщується в однобайтовій команді разом з кодом операції.

Як приклад виконання команди МП К580ВМ80А непрямого завантаження в акумулятор вмісту комірки пам'яті з адресою 0012_{16}, що зберігається в регістровій парі DE, показано на рис. 1.10

Команда непрямого завантаження акумулятора є однобайтовою і крім коду операції містить номер 01 регістрової пари

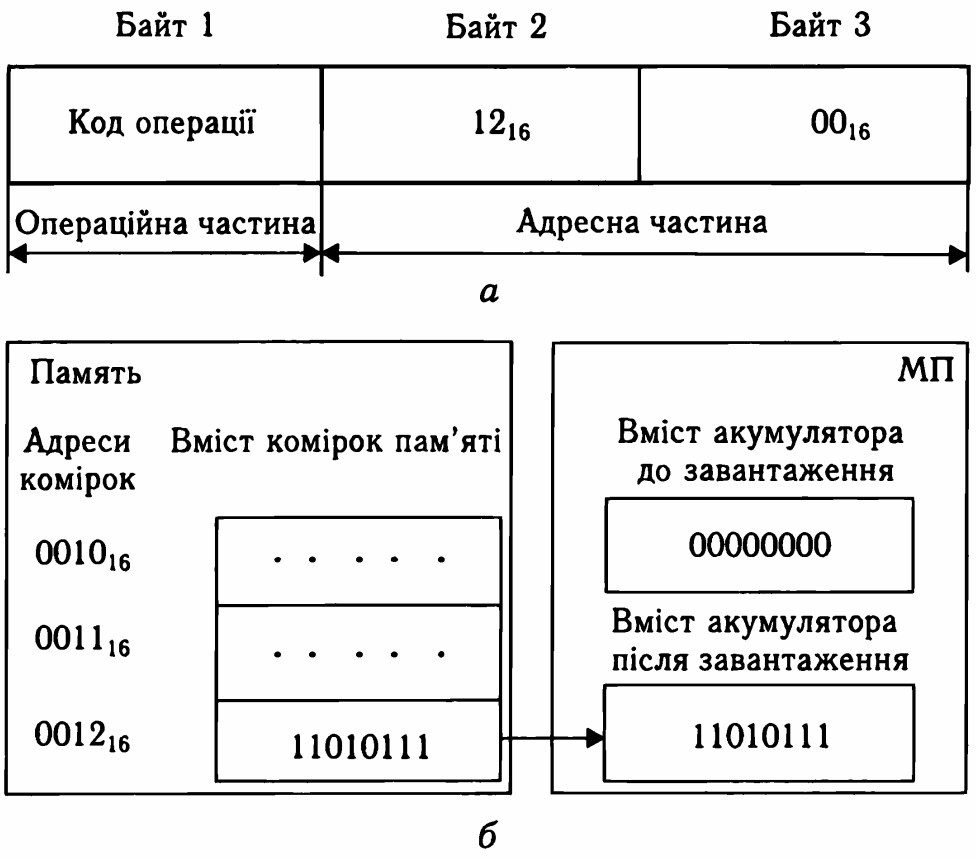


Рис. 1.9. Команда прямого завантаження в акумулятор вмісту комірки пам'яті:
а — формат команди; *б* — схема виконання

DE. Старша частина адреси комірки пам'яті (00_{16}) зберігається у регістрі *D*, а молодша частина (12_{16}) — у регістрі *E*.

Вміст регістрової пари передається в регістр адреси МП, внаслідок чого вміст 11010111_2 комірки з адресою 0012_{16} копіюється в акумуляторі.

Безпосередня адресація. У першому байті команди з безпосередньою адресацією розміщується код операції. Значення операндів заносяться у команду під час програмування і знаходяться у другому або другому і третьому байтах.

Цими значеннями зазвичай є деякі константи, заздалегідь відомі програмісту. У процесі виконання програми значення операндів залишаються незмінними, оскільки вони разом з командою розміщуються в постійному запам'ятовувальному пристрої (ПЗП). Використання такого способу не потребує адрес операндів. Як приклад на рис. 1.11 зображено формат і схему виконання команди безпосереднього завантаження акумулятора значенням 11010111_2, яке зберігається у другому байті команди. Після виконання команди це число копіюється в акумуляторі. Після кожного чергового звернення до цієї команди в акумулятор записується таке саме число.

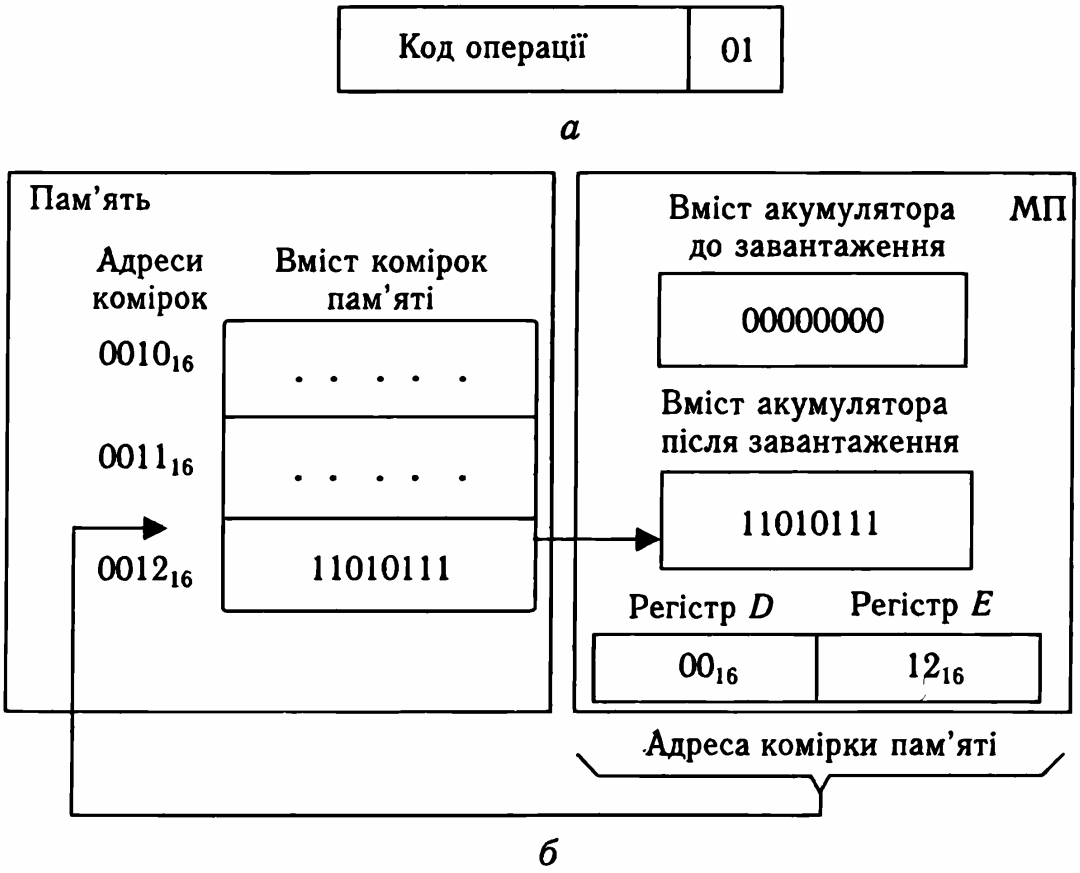
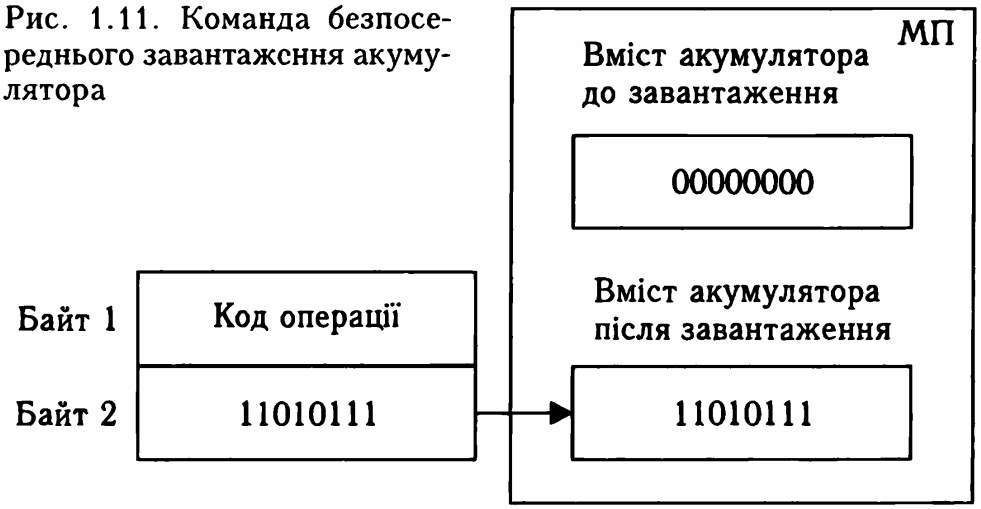


Рис. 1.10. Команда непрямого завантаження акумулятора:
а — формат команди; *б* — схема виконання

Рис. 1.11. Команда безпосереднього завантаження акумулятора



Автоінкрементна (автодекрементна) адресація. У процесі автоінкрементної адресації адреса операнда обчислюється так само, як і за непрямої адресації, а потім здійснюється збільшення вмісту регістра: на один — для звернення до наступного байта, на два — для звернення до наступного слова. Розмір операнда визначається кодом операції.

Сторінкова адресація. Під час використання сторінкової адресації пам'ять поділяється на кілька сторінок однакової довжини. Адресація сторінок здійснюється або з програмного лічильника, або з окремого регістра сторінок. Адресація пам'яті всередині сторінок здійснюється адресою, що міститься в команді.

Індексна адресація. Для утворення адреси операнда до значення адресного поля команди додають значення вмісту індексного регістра, яке називають *індексом.*

Відносна адресація. За відносної адресації адреса операнда визначається додаванням вмісту програмованого лічильника або іншого регістра із зазначеним у команді числом. Вміст програмованого лічильника або іншого регістра називають *базовою адресою.* Для збереження базових адрес у МП можуть бути передбачені *базові регістри* або спеціально виділені комірки пам'яті. Тоді в адресному полі команди вказується номер базового регістра.

У МПС використовується програмування мовою Асемблер. ***Асемблером*** називається і мова програмування у мнемокодах команд, і спеціальна програма-транслятор, що переводить (транслює) мнемокоди на машинні коди, які зчитуються мікропроцесором з пам'яті програм, дешифруються і виконуються. Процес переведення мнемокодів на машинні коди називають *асемблюванням.*

Програма на мові Асемблер містить два типи *виразів*:

- команди, що транслюються на машинні коди;
- директиви, що керують ходом трансляції.

Вираз має такий вигляд:

{<мітка>}: <мнемокод> {<операнд>} {,} {< операнд >} {; коментар}.

У фігурних дужках наведено елементи виразу, яких може не бути у деяких командах. Мітка, мнемокод і операнди відокремлюються хоча б одним пробілом або табуляцією. Максимальна довжина рядка становить 132 символи, але найчастіше використовуються рядки з 80 символів, що відповідає довжині екрана.

Прикладами *команд* мови Асемблер є:

```
Мітка    Мнемокод  Операнд(и)  Коментар
         MOV       AX, 0       ; Команда, два операнди
M1:      ADD       AX, BX      ; Мітка, команда, два опе-
                               ; ранди
DELAY:   MOV       CX, 1234    ; Мітка, команда, два опе-
                               ; ранди
```

Прикладом *директиви* є:

```
Мітка     Мнемокод  Операнд(и)   Коментар
COUNT:    DB        1            ; Мітка, команда, один
                                 ; операнд
```

Мітка на мові Асемблер є символічною адресою команди. Мітками позначаються не всі команди, а лише ті, до яких треба виконувати перехід за допомогою команд переходів або викликів підпрограм. У командах переходів або викликів підпрограм позначення мітки використовується як операнд — символічна адреса переходу, наприклад:

```
Мітка     Мнемокод  Операнд(и)   Коментарі
          JMP       M1           ; Перехід до команди з
                                 ; міткою M1
          CALL      DELAY        ; Виклик підпрограми з
                                 ; міткою DELAY.
```

Після мітки ставиться двокрапка. Першим символом у мітці має бути літера або один із спеціальних символів: знак питання «?»; крапка «.»; знак амперсанд «@»; підкреслювання «_»; знак долара «$». Знак питання і крапка можуть займати лише перше місце. Максимальна довжина мітки — 31 символ. Приклади міток: *COUNT*, *PAGE*25, $*E*10. Рекомендується використовувати описові та смислові мітки. Усі мітки у програмі мають бути унікальними, тобто не може бути кількох команд з однаковими мітками. Крім того, як мітки не можна використовувати зарезервовані асемблером слова, до яких належать коди команд, директиви, імена регістрів. Наприклад, імена *AX*, *DI* та *AL* є зарезервованими і використовуються лише для зазначення відповідних регістрів.

Мнемокод ідентифікує команду асемблера. Для мнемокодів використовують скорочені або повні англійські слова, які передають значення основної функції команди: *ADD* — додати, *SUB* (*SUBtract*) — відняти, *XCHG* (*eXCHanGe*) — поміняти.

Операнди відокремлюються комами. Якщо задано два операнди, то перший з них завжди є приймачем, а другий — джерелом інформації. Команда може містити різну кількість операндів різних типів, наприклад:

```
Мітка  Мнемокод  Операнд(и)   Коментарі
       RET                    ; Повернутися (операнди не
                              ; вказані)
       INC       CX           ; Збільшити CX (один операнд)
       ADD       AX, 12H      ; Додати 12H до вмісту AX
                              ; (два операнди)
       MOV       BX, [SI]     ; Занести до регістра BX чис-
                              ; ло з комірки
                              ; Пам'яті з адресою DS:SI
                              ; (два операнди)
```

Коментарі ігноруються у процесі трансляції і використовуються для документування та кращого розуміння змісту програми. Коментар завжди починається із символу «;» і може містити будь-які символи. Коментар також може займати увесь рядок або бути розташованим за командою в одному рядку, наприклад:

```
Мітка  Мнемокод  Операнд(и)   Коментарі
                              ; Цей рядок є коментар
                              ; ем
       ADD       AX, BX       ; Команда і коментар в
                              ; одному рядку
```

Оскільки коментарі не транслюються на машинні коди, то їхня кількість не впливає на ефективність виконання програми.

Програма мовою Асемблер називається *початковою програмою*, або *початковим програмним модулем.* Асемблювання, або перетворення початкової програми на машинні коди, виконує програма-транслятор, наприклад *TASM.COM*. Залежно від установок, які задає користувач, програма переводить початковий модуль в один з двох програмних модулів: командний модуль (файл з розширенням *.COM*) або об'єктний модуль (файл з розширенням *.OBJ*).

Командний модуль містить машинні коди команд з абсолютними адресами і виконується МП. Командний модуль доцільно використовувати тоді, коли ємність програми не перевищує розміру одного сегмента (64 Кбайт). Першим оператором командного модуля є директива ORG 100H (*ORIGIN* — початок), яка розташовує першу команду програми у сегменті кодів зі зміщенням 100*H*. Закінчуватися програма має або командою RET, або стандартною процедурою коректного виходу до *MSDOS*:

```
Мітка  Мнемокод  Операнд(и)   Коментарі
       MOV       AH, 4CH      ; Занести в AH число 4CH
                              ; (значення параметра пе-
                              ; реривання INT 21H)
       INT       21H          ; Викликати стандартну
                              ; процедуру
                              ; переривання 21H-коректного
                              ; виходу до MSDOS.
```

Останнім записом програми має бути директива *END*.

Об'єктний модуль містить машинні коди команд з відносними адресами. Об'єктний модуль виконується МП після заміни відносних адрес на абсолютні за допомогою програми-укладача, наприклад *LINK.EXE*, яка генерує модуль з розширенням *.EXE* (*EXE*-файл або *EXE*-програму);

EXE-файл, на відміну від командного модуля, може перевищувати ємність одного сегмента. Однак у цьому разі обов'язковим є визначення сегментів за допомогою директив асемблера. Закінчується *EXE*-файл стандартною процедурою коректного виходу до *MSDOS*.

Програма-укладач має ще одне призначення. Вона об'єднує об'єктний модуль з бібліотечними модулями або кілька окремих об'єктних модулів об'єднує в один *EXE*-файл. *Бібліотечними модулями* називають об'єктні файли, що містять найпоширеніші підпрограми. Бібліотечні модулі розміщуються у спеціальному системному файлі — бібліотеці (*LIBRARY*).

Під час асемблювання програма-транслятор генерує лістинг і файл лістингу програми. *Лістинг* — відображення на дисплеї або папері текстів початкового програмного модуля, програмного модуля (*.COM* або *.OBJ*) та повідомлень, які вказують на помилки програмування, зумовлені порушенням правил запису виразів, наприклад немає операнда або неправильний мнемокод команди.

Директиви призначені для керування процесом асемблювання і формування лістингу. Вони діють лише у процесі асемблювання програми і не переводяться на машинні коди. Мова Асемблер містить такі основні директиви:

- початок і кінець сегмента *SEGMENT* та *ENDS*;
- початок і кінець процедури *PROC* та *ENDP*;
- призначення сегментів *ASSUME*;
- початок *ORG*;
- розподіл та ініціювання пам'яті *DB*, *DW*, *DD*;
- завершення програми *END*;
- відзначення *LABEL*.

Директиви початку і кінця сегмента SEGMENT та ***ENDS*** призначені для опису сегментів, які використовує програма. Директиви початку і кінця сегмента використовують разом, наприклад:

Назва	Мнемокод	Операнд
DATASG	*SEGMENT*	{<параметри>}
	•	
	•	Інші команди або директиви сегмента
	•	
DATASG	*ENDS*	

Обидві директиви *SEGMENT* і *ENDS* повинні мати однакові назви. Директива *SEGMENT* може містити три типи параметрів: вирівнювання, об'єднання і класу.

Параметр вирівнювання визначає початкову адресу, або *межу,* сегмента, наприклад:

PAGE = xxx00;
PARA = xxxx0 (межа за замовчуванням);
WORD = xxxxe (парна межа);
BYTE = xxxxx,

де x — будь-яка шістнадцяткова цифра; e — парна шістнадцяткова цифра. Якщо немає параметра вирівнювання за замовчуванням, береться параметр *PARA*, який вказує на те, що сегмент розташовується на початку параграфа, а початкова адреса сегмента є кратною 16. *Параграфом* називають область пам'яті розміром 16 байт, початкова адреса якої кратна 16, тобто має праворуч чотири нульових розряди.

Параметр об'єднання вказує на спосіб обробки сегмента під час компонування кількох програмних модулів:

NONE: значення за замовчуванням. Сегмент має бути логічно відокремленим від інших сегментів, хоча фізично він може розташовуватися поряд. Передбачається, що сегмент має власну базову адресу;

PUBLIC: усі *PUBLIC* — сегменти з однаковими назвою і класом завантажуються у суміжні області та мають одну базову адресу;

STACK: призначення аналогічне параметру *PUBLIC*. У будь-якій програмі має бути визначений принаймні один сегмент *STACK*. Якщо визначено більше одного стеку, то покажчик стеку *SP* (*Stack Pointer*) встановлюється на початок першого стеку;

COMMON: для сегментів *COMMON* з однаковими назвами та класами встановлюється одна спільна базова адреса. Під час виконання програми здійснюється накладання другого сегмента на перший. Розмір загальної області визначається найдовшим сегментом;

AT-параграф: цей параметр забезпечує визначення міток та змінних за фіксованими адресами у фіксованих областях пам'яті;

'Клас': цей параметр може мати будь-яку правильну назву, яка розміщується в одинарних лапках. Параметр використовується для обробки сегментів, що мають однакові назви та класи. Типовими є класи *'STACK'* та *'CODE'*, наприклад:

Назва	Мнемокод	Операнд
STACKSG	SEGMENT	PARA STACK 'STACK'

Якщо програма не повинна об'єднуватися з іншими програмами, параметр об'єднання не вказується.

Директиви початку і кінця процедури PROC та ***ENDP*** використовуються для визначення підпрограм у сег-

менті кодів і мають такий формат:

<Назва> *PROC* {<тип процедури>}.

Можливі два типи процедур:

• *NEAR* — процедура знаходиться в тому самому сегменті, що й команди, які її викликають;

• *FAR* — процедура знаходиться за межами сегмента.

За замовчуванням береться тип процедури *NEAR*.

Сегмент кодів може містити кілька процедур. Описання сегмента кодів, що містить лише одну процедуру, має такий вигляд:

Назва	Мнемокод	Операнд
Ім'я_сегмента	*SEGMENT*	*PARA*
Ім'я_процедури	*PROC*	*FAR*
		RET
Ім'я_процедури	*ENDP*	
Ім'я_сегмента	*ENDS*	

Ім'я процедури має бути обов'язково і збігатися з іменем у директиві *ENDP*, яка визначає кінець процедури.

Директива призначення сегментів ASSUME використовується для встановлення відповідності між сегментами та сегментними регістрами і має такий формат:

ASSUME <сегментний регістр>:<ім'я>{,}{...}.

Наприклад, запис *SS*:ім_стек вказує, що ім'я стеку визначається вмістом регістра *SS*. Одна директива *ASSUME* може призначати до чотирьох сегментних регістрів у будь-якій послідовності, наприклад:

Мнемокод	Операнд(и)
ASSUME	*SS*:ім_стек, *DS*:ім_дані, *CS*:ім_код, *ES*:ім_додатковий_дані

Для скасування будь-якого призначеного раніше у директиві *ASSUME* сегментного регістра треба використовувати слово *NOTHING*:

Мнемокод	Операнд(и)
ASSUME	*ES*: *NOTHING*.

Якщо програма не використовує будь-який сегмент, то відповідний йому операнд можна відпустити або вказати слово *NOTHING*.

Директива ORG використовується для зміни вмісту програмованого лічильника без команд умовного чи безумовного переходу. Найчастіше цю директиву використовують для встановлення початкової адреси програми, наприклад директива *ORG* 100*H* встановлює програмований лічильник на

зміщення 100H відносно початку сегмента кодів. Операнд зі знаком долара «$» має поточне значення програмованого лічильника, наприклад директива *ORG* $+10*H* збільшує адресу, завантажену у програмований лічильник, на 10*H*.

Директиви розподілу та ініціювання пам'яті використовуються для визначення вмісту та резервування комірок пам'яті.

Директива має формат:

{ <ім'я> } *Dn* {кількість повторень *DUP*}<вираз>,

де мнемокод $Dn = \begin{Bmatrix} DB \\ DW \\ DD \\ DQ \\ DT \end{Bmatrix}$ вказує на довжину даних: *DB* — байт; *DW* — слово (два байти); *DD* — подвійне слово; *DQ* — чотири слова; *DT* — десять байтів. Якщо у форматі наявне ім'я, то далі у програмі воно може використовуватися для позначення комірки пам'яті.

<Вираз> у форматі директиви містить одну або кілька констант для задання початкових значень вмісту комірок пам'яті або знак «?» для невизначеного значення вмісту. Наприклад, директива

ALPHA DB 34

означає, що комірка пам'яті з іменем *ALPHA* містить число 34. У ході виконання програми вміст комірки може бути змінений. Директива

BETA DW ?

визначає, що комірка з іменем *BETA* має розрядність 16, але вміст комірки є невизначеним. Директива може містити кілька констант, розділених комами і обмежених лише довжиною рядка. Наприклад, вираз

ARRAY DB 01, 02, 11, 12, 21, 22

визначає 6 констант у вигляді послідовності суміжних байтів. Посилання на комірку з іменем *ARRAY* вказує на першу константу (01), з іменем *ARRAY* + 1 — на другу (02), з іменем *ARRAY* + 2 — на третю (11) і т. д. Запис *MOV AL*, *ARRAY* + 4 завантажує у регістр *AL* значення 21.

Одна директива може визначити кілька комірок пам'яті. У цьому разі директива має вигляд:

{ <ім'я> } *Dn* {кількість повторень} *DUP* <вираз>.

Наприклад, директива, що визначає 5 байт, які містять число 21, записується так:

DB 5 *DUP* (21)

Директива завершення програми END є останньою у програмі та має такий формат:

END {<стартова адреса>}.

Параметр директиви <стартова адреса> використовують лише для створення *EXE*-файлів.

Директива відзначення LABEL призначена для встановлення відповідності між іменем і типом змінних. Вона має такий формат:

<ім'я> *LABEL* {<тип>}.

Як тип можна використовувати слова *BYTE, WORD, DWORD,* що визначають довжину даних: байт, слово або подвійне слово. Директива *LABEL* перевизначає параметри процедур *NEAR* або *FAR*. Наприклад, директива

TOS LABEL WORD

присвоює комірці пам'яті ім'я *TOS* і зазначає, що її вміст є словом.

Приклади написання простих програм. Прості програми доцільно оформляти у вигляді командних файлів. Першою директивою таких програм є директива *ORG* 100*H*, останньою — *END*.

Приклад 1.1. Написати програму додавання вмісту двох 8-розрядних комірок пам'яті, що знаходяться в сегменті даних *DS* зі зміщеннями 1000*H* і 1001*H*. Результат розмістити у комірці пам'яті з адресою *DS* : 1002*H*.

У цьому прикладі для простоти не будемо враховувати можливість виникнення перенесення. Програма матиме такий вигляд:

```
Мнемокод    Операнд(и)      Коментарі
ORG         100H            ; Початок програми
MOV         AL, [1000H]     ; AL ← DS:[1000H]
                            ; Переслати у 8-розрядний регістр AL
                            ; вміст комірки
                            ; пам'яті з адресою DS:1000H
ADD         AL, [1000H]     ; AL ← AL + DS:[1000H]
                            ; Додати до вмісту AL вміст комірки
                            ; DS:[1001H]
MOV         [1002H], AL     ; DS:[1002H] ← AL
                            ; Переслати вміст AL у комірку DS:[1002H]
END                         ; Завершення програми
```

Зазначимо, що запис *MOV AL*, [1000*H*] рівнозначний запису *MOV AL*, *DS*:[1000*H*], оскільки сегмент *DS* прийнятий за замовчуванням.

Приклад 1.2. Написати програму, яка забезпечує розподіл вмісту 16-розрядної комірки пам'яті з адресою *ES*:[2000*H*] на чотири тетради. Тетради мають бути записані у молодші частини чотирьох послідовних 8-розрядних комірок пам'яті, починаючи з адреси *DS*:1000*H*, причому старшу тетраду треба записати у комірку зі старшою адресою.

У цьому прикладі для запису результату краще використовувати непряму адресацію. Програма має такий вигляд:

Мнемокод	Операнд(и)	Коментарі
ORG	100*H*	
MOV	*AX, ES*:[2000*H*]	; *AX* ← *ES*:[2000*H*]
		; Переслати вміст 16-розрядної комір-
		; ки *ES*:[2000*H*]
		; у 16-розрядний регістр *AX*
MOV	*DX, AX*	; *DX* ← *AX*, зберегти початкове число
		; в *DX*
AND	*AX*, 000*FH*	; *AX*←*AX* ∧ 0000 0000 0000 1111
		; Виділити молодшу тетраду (скинути
		; всі розряди
		; AX, крім чотирьох молодших)
MOV	*SI*, 1000*H*	; *SI* ← 1000*H*
		; Записати в *SI* початкову адресу ре-
		; зультату
MOV	[*SI*], *AL*	; *DS*:[*SI*] ← *AL*
		; Переслати вміст комірки пам'яті з
		; адресою *DS:SI* у
		; молодшу 8-розрядну частину *AL* ре-
		; гістра *AX*
MOV	*AX, DX*	; *AX* ← *DX*
		; Переслати початкове число з *DX* у *AX*
AND	*AX*, 00F0*H*	; *AX*←*AX* ∧ 0000 0000 1111 0000
		; Виділити другу тетраду
MOV	*CL*, 4	; Завантажити у *CL* число розрядів зсуву
ROR	*AL, CL*	; Циклічний зсув *AL* на чотири розряди
		; праворуч,
		; унаслідок чого виділене чотирирозряд-
		; не число
		; переміститься в *AL*
INC	*SI*	; *SI* ← *SI* + 1
		; Збільшити *SI* для запису другого числа
MOV	[*SI*], *AL*	; *DS*:[*SI*] ← *AL*, запам'ятати другу тет-
		; раду
		; у комірці *DS*:[*SI*]
MOV	*AX, DX*	; *AX* ← *DX*, переслати початкове число
		; з *DX* у *AX*
AND	*AX*, 0F00*H*	; *AX* ← *AX* ∧ 0000 1111 0000 0000,
		; Виділити третю тетраду
INC	*SI*	; *SI* ← *SI* + 1, збільшити адресу резуль-
		; тату
MOV	[*SI*], *AH*	; *DS*:[*SI*] ← *AH*, запам'ятати третю тет-
		; раду

```
MOV     AX, DX          ; AX←DX, переслати вихідне число з DX
                        ; у AX
AND     AX, 0F000H      ; AX ← AX ∧ 1111 0000 0000 0000,
                        ; Виділити четверту тетраду
INC     SI              ; Збільшити адресу результату
MOV     CL, 4           ; Завантажити у CL число розрядів
ROR     AH, CL          ; Циклічний зсув AL на чотири розряди
                        ; праворуч
MOV     [SI], AH        ; Запам'ятати четверту тетраду
END.
```

Для зменшення громіздкості програми, її доцільно зводити до однотипних кроків і використовувати циклічні операції. Розглянутий приклад можна спростити, якщо виконати зсув 16-розрядного числа так, щоб тетрада, яка виділяється, завжди була молодшою. Алгоритм такої програми разом з командами зображено на рис. 1.12.

Типові обчислювальні процедури. Алгоритм типової обчислювальної процедури ЯКЩО-ТО-ІНАКШЕ показано на рис. 1.13.

Цю процедуру застосовують тоді, коли треба реалізувати перехід до однієї з двох обчислювальних процедур залежно від умови. Написання програм мовою Асемблер виконується командами переходів за умовами встановлення (скидання) прапорців.

Приклад 1.3. Написати програму ділення вмісту *AX* на вміст *BL*. Результат помістити у 8-розрядну комірку пам'яті з адресою *DS* : 1000*H*. Остачею від ділення знехтувати. Якщо вміст *BL* = 0, то ділення не виконувати, на місце результату помістити число 0*FFH*.

Програма має такий вигляд:

```
Мітка   Мнемокод   Операнд(и)      Коментарі
        ORG        100H
        CMP        BL,0            ; Порівняти вміст BL з нулем;
                                   результат
                                   ; команди впливає на встанов-
                                   ; лення прапорця
                                   ; нуля Z
        JZ         M1              ; Якщо Z = 1 (BL = 0), то пе-
                                   ; рехід на мітку M1,
        DIV        BL              ; інакше виконати ділення AL ←
                                   ; AX : BL,
                                   ; остача → AH
        JMP        M2              ; Безумовний перехід на мітку M2
M1:     MOV        AL,0FFH         ; Занести число 0FFH у AL
M2:     MOV        [1000H], AL     ; Запам'ятати результат у ко-
                                   ; мірці DS : [1000H]
        END.
```

Процедура ЯКЩО-ТО (рис. 1.14) є частковим випадком процедури ЯКЩО-ТО-ІНАКШЕ і використовується тоді, коли

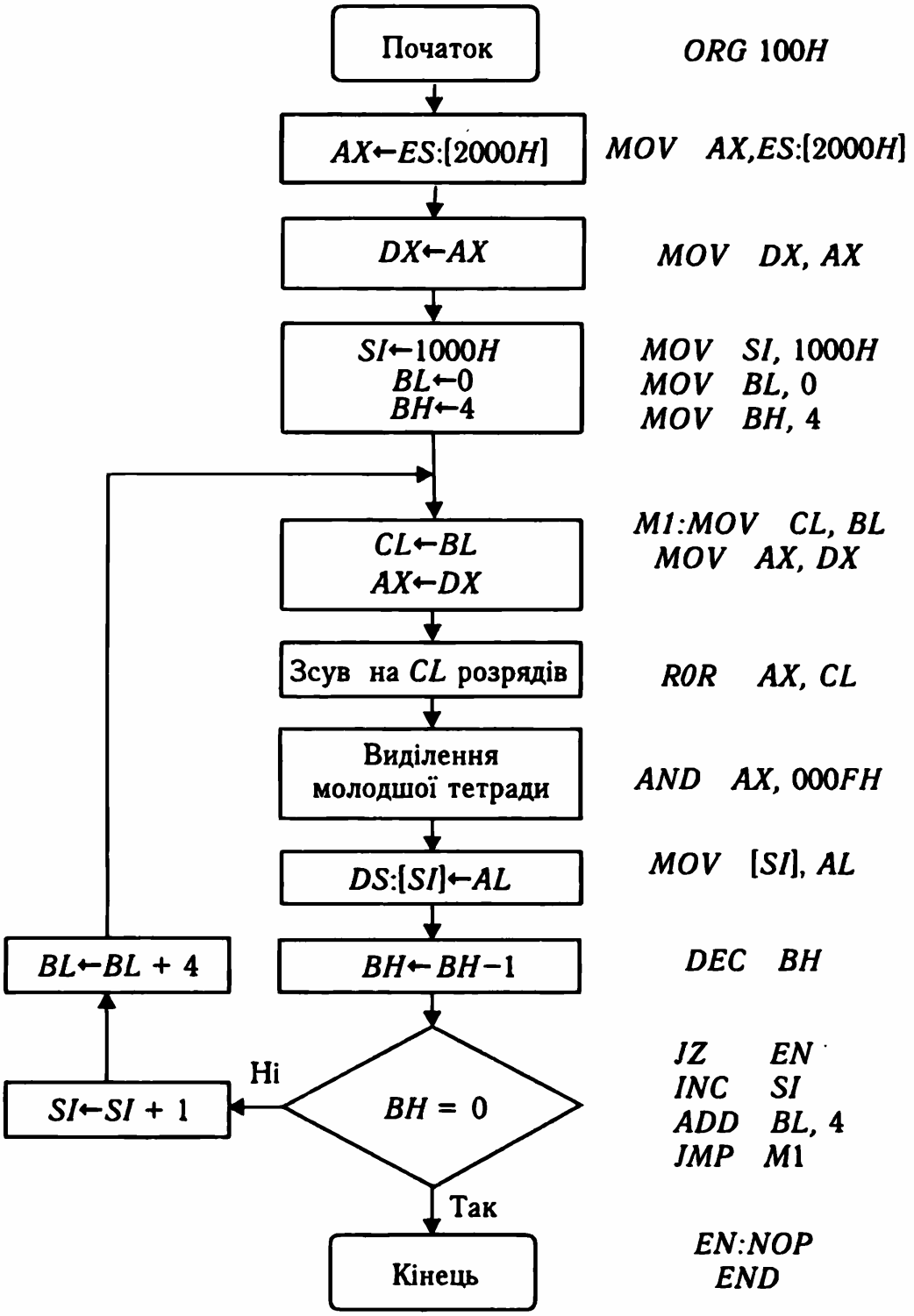


Рис. 1.12. Алгоритм розв'язання задачі прикладу 1.2

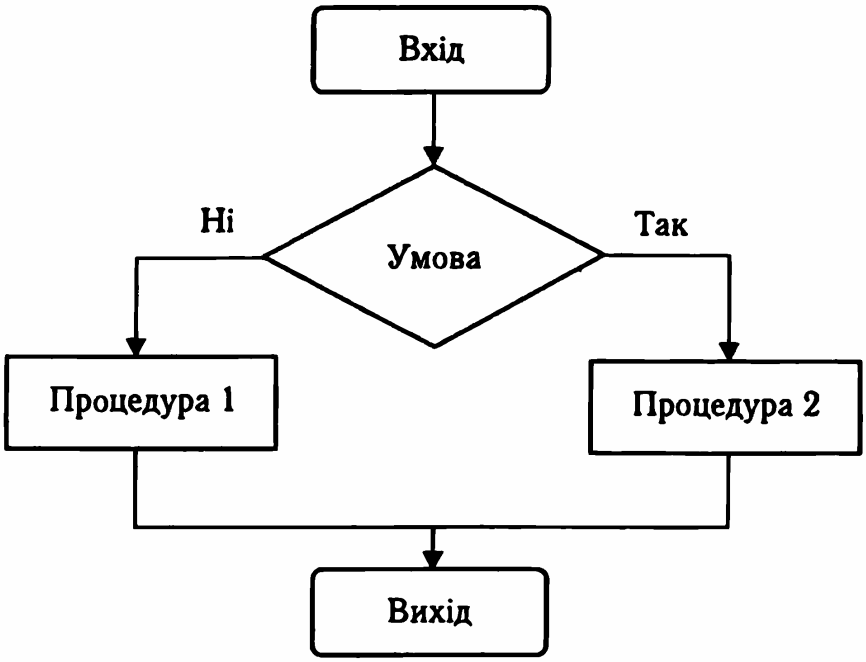


Рис. 1.13. Алгоритм процедури ЯКЩО-ТО-ІНАКШЕ

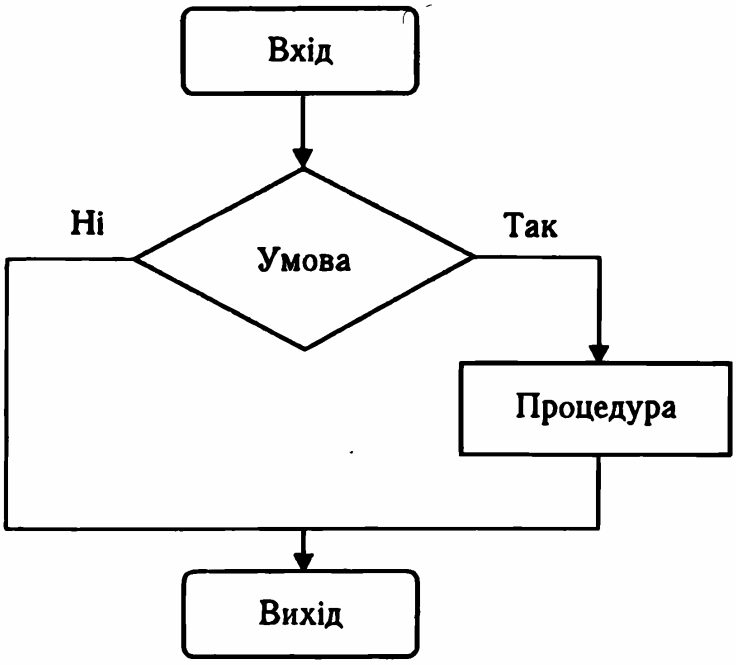


Рис. 1.14. Алгоритм процедури ЯКЩО-ТО

треба реалізувати одну обчислювальну процедуру залежно від умови.

Процедуру РОБИ-ПОКИ (рис. 1.15) використовують для повторення однотипних дій до моменту виконання умови закінчення циклу.

Приклад 1.4. Написати програму додавання за модулем 256 масиву з 100*H* байт, розміщених за початковою адресою 7000*H*:3000*H*.

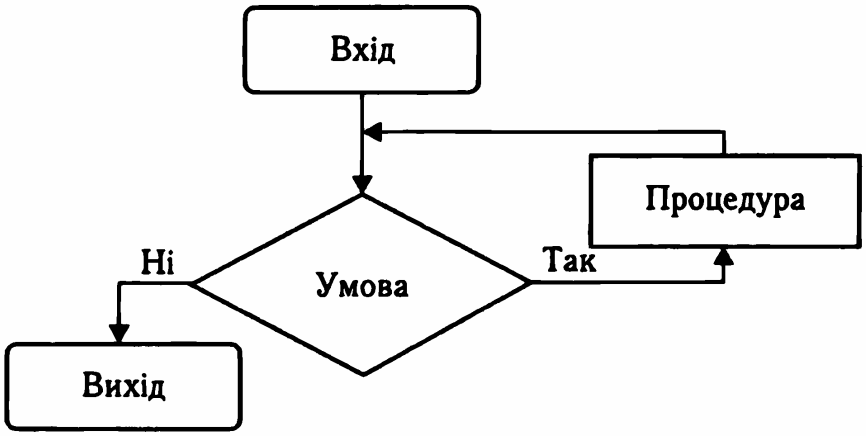


Рис. 1.15. Алгоритм процедури РОБИ-ПОКИ

Рсзультат у вигляді одного байта записати у комірку з адресою 7000*H*:5000*H*.

Програма має такий вигляд:

Мітка	Мнемокод	Операнд(и)	Коментарі
	ORG	100*H*	; Початок
	MOV	*AX*, 7000*H*	; Завантажити в *AX* адресу сегмента
	MOV	*DS*, *AX*	; Завантажити в *DS* адресу сегмента
	MOV	*SI*, 3000*H*	; Завантажити в *SI* зміщення першого ; елемента масиву
	MOV	*CX*, 101*H*	; Завантажити у лічильник *CX* довжину ; масиву +1
	MOV	*AL*, [*SI*]	; Завантажити у *AL* перший елемент масиву
*M*1:	*LOOP*	*M*0	; Зменшити вміст *CX* на 1, якщо ; *CX* ≠ 0, то ; перейти на мітку *M*0,
	MOV	[5000*H*], *AL*	; інакше запам'ятати результат у ; *DS*:[5000*H*]
	JMP	*EXIT*	; вихід
*M*0:	*INC*	*SI*	; *SI* ← *SI* + 1 — адреса наступного ; елемента
	ADD	*AL*, [*SI*]	; Додати вміст *DS*:*SI* до попередньої суми ; в акумулятори *AL*
	JMP	*M*1	; Псрейти на мітку *M*1 для перевірки умови ; виходу з циклу
	END.		

Процедура ПОВТОРЮЙ-ДО-ТОГО-ЯК (рис. 1.16) аналогічна попередній, але однотипні дії виконуються перед перевіркою умови.

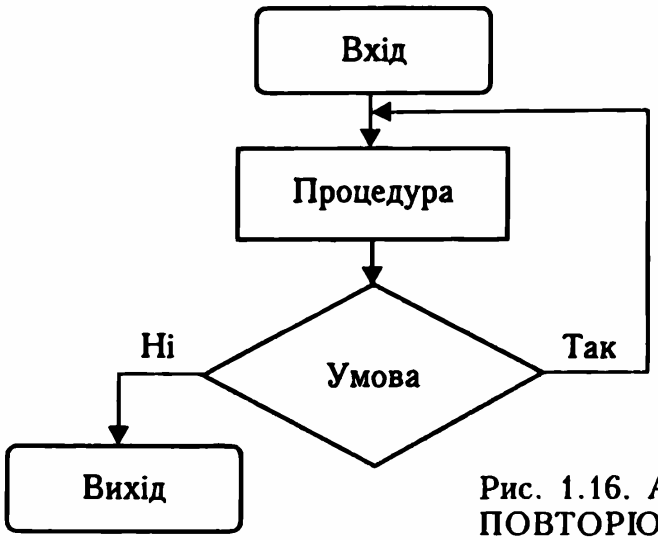


Рис. 1.16. Алгоритм процедури ПОВТОРЮЙ-ДО-ТОГО-ЯК

Програма виконання завдання прикладу 1.4 згідно з алгоритмом (1.21) має такий вигляд:

```
Мітка  Мнемокод  Операнд(и)   Коментарі
       ORG       100H         ; Початок
       MOV       AX,7000H     ; Завантажити в AX адресу сегмен-
                              ; та
       MOV       DS,AX        ; Завантажити в DS адресу сегмента
       MOV       SI,3000H     ; Завантажити в SI зміщення пер-
                              ; шого
                              ; елсмента масиву
       MOV       CX,100H      ; Завантажити в лічильник CX дов-
                              ; жину
                              ; масиву +1
       MOV       AL,[SI]      ; Завантажити в AL перший елемент
                              ; масиву
M0:    INC       SI           ; SI←SI+1 — адреса наступного еле-
                              ; мента
       ADD       AL,[SI]      ; Додати вміст DS:SI до попередньої
                              ; суми в AL
       LOOP      M0           ; Зменшити вміст CX на одиницю,
                              ; якщо CX ≠ 0, то перейти до мітки
                              ; M0,
       MOV       [5000H],AL   ; інакше запам'ятати результат
                              ; у DS:[5000H]
       JMP       EXIT         ; вихід
       END.
```

Написання *EXE*-програм. Написання *EXE*-програм має виконуватися за таких умов:

- зазначення відповідності між сегментами та сегментними регістрами;
- зберігання вмісту *DS* у стеку;

• записування числа 0 у стек;
• завантаження адреси сегмента даних у регістр *DS*.

Перша вимога виконується за допомогою директиви *ASSUME*, інші — за допомогою відповідних команд асемблера.

Приклад 1.5. Написати EXE-програму знаходження максимального числа у масиві 8-розрядних беззнакових чисел. Результат записати в регістр *DL*.

Програма має такий вигляд:

```
Мітка     Мнемокод   Операнд(и)        Коментарі
(або ім'я)
DATASG    SEGMENT    PARA 'DATA'       ; Визначити сегмент даних
MASSIV    DB         01,02,03,45,      ; Визначити у сегменті
                     56,67,78,89,      ; даних 10 значень масиву
                     0FE,10            ; з іменем MASSIV
DATASG    ENDS
STACKSG   SEGMENT    PARA STACK        ; Визначити сегмент стеку
                     'Stack'
          DW         100 DUP (?)       ; Визначити 100 слів
TOS       LABEL      WORD              ; Визначити ім'я і формат
                                       ; Вершини стеку
STACKSG   ENDS
CODESG    SEGMENT    PARA 'CODE'       ; Визначити сегмент кодів
          BEGIN      PROC FAR          ; Початок процедури
          ASSUME     SS: STACKSG,
                     DS:DATASG,
                     CS: CODESG
          PUSH       DS                ; Завантажити вміст DS
                                       ; у стек
          SUB        AX,AX             ; Встановити нульовий вміст
                                       ; у AX
          PUSH       AX                ; Записати нуль у стек
          MOV        AX,DATASG         ; Завантажити адресу DATASG
                                       ; у AX
          MOV        DS,AX             ; Записати адресу DATASG
                                       ; у регістр DS
          LEA        BX,MASSIV         ; Завантажити у регістр BX
                                       ; адресу першого елемента
                                       ; масиву
          MOV        CX,10             ; Завантажити у CX довжину
                                       ; масиву
          MOV        DL,[BX]           ; DL←DS:[BX]
COMP:     MOV        AL,[BX]           ; AL←DS:[BX]
          CMP        AL,[BX+1]         ; Порівняти два сусідніх
                                       ; елементи масиву
          JAE        NEXT              ; Якщо вміст попереднього
                                       ; елемента масиву [BX] біль-
                                       ; ший
                                       ; або дорівнює вмісту
```

```
                                        ; наступного [BX +1], то пе-
                                        ; рейти
                                        ; на мітку NEXT,
        MOV     DL,[BX+1]               ; інакше завантажити у DL
                                        ; значення [BX + 1]
NEXT:   INC     BX                      ; збільшити BX для адре-
                                        ; сації
                                        ; наступного елемента масиву
        LOOP    COMP                    ; перевірка умови виходу з
                                        ; циклу
        RET                             ; повернення
        BEGIN   ENDP                    ; кінець процедури BEGIN
CODESG  ENDS                            ; кінець сегмента кодів
        END.                            ; кінець програми
```

У розглянутому прикладі вихід у *MS DOS* здійснюється командою *RET* з використанням для цього адреси, записаної у стек на початку програми командою *PUSH DS*. Інакше можна завершити програму командою *INT 20H*.

Контрольні запитання

1. Назвіть складові МПК.
2. За якими класифікаційними ознаками поділяють МП і МПК?
3. На розв'язання яких задач орієнтовані спеціалізовані МП?
4. Які переваги і недоліки мають секційні МП порівняно з однокристальними?
5. Яке призначення та які складові системної шини?
6. Назвіть принципи передавання інформації по шинах — адреси, даних, керування.
7. Назвіть принципи побудови МПС і схарактеризуйте їх.
8. Наведіть типову структуру МПС і поясніть призначення функціональних модулів.
9. Поясніть призначення входу керування третім станом.
10. Дайте визначення архітектури МП.
11. Яка відмінність між гарвардською і фоннеймановською архітектурами?
12. Які функції виконує пристрій керування?
13. Що визначає вміст лічильника команд та як він змінюється?
14. Чим відрізняється акумулятор від інших регістрів МП?
15. Які основні характеристики та структурні схеми процесорів з фоннеймановською і гарвардською архітектурою?
16. Яке призначення регістрів МП з фоннеймановською архітектурою?
17. Схарактеризуйте функції пристрою керування.
18. Наведіть визначення та призначення акумулятора.
19. Поясніть роботу програмованого лічильника.
20. Укажіть місцезнаходження операнда з прямою адресацією.
21. Як визначити адресу операнда з непрямою адресацією?

22. Як визначити значення операнда з безпосередньою адресацією?
23. Як визначити адресу операнда з відносною адресацією?
24. Напишіть програму пересилання вмісту 8-розрядної комірки пам'яті з адресою 7000*H*:1000*H* у 8-розрядний регістр *AL*.
25. Напишіть програму віднімання вмісту двох послідовних комірок пам'яті з адресами *DS*:35*A*0*H* і 35*A*1*H* із записом результату в комірку з адресою 35*A*2*H*.
26. Напишіть програму додавання масивів байтів з адресами 8350:4735*H* і 3660:2200*H* за правилом «перший з першим, другий з другим і т. д.». Занести в масив 6250:2400*H* адреси тих пар доданків, сума яких дорівнює нулю. Довжина масиву 100*H*.
27. Виконайте ділення масиву з 25*H* слів 5*B*00:3000*H* на масив з 25*H* байт 5*C*00:4000*H* за правилом «перший на перший, другий на другий і т.д.». Результати занести в масив 6000:5000*H*. За потреби ділення на 0 ділення не проводити, а байти результату завантажити числом 1*AH*.

Розділ 2 ОДНОКРИСТАЛЬНІ МІКРОПРОЦЕСОРИ

2.1. Однокристальний 8-розрядний мікропроцесор

Структурну схему узагальненого 8-розрядного однокристального МП зображено на рис. 2.1. Схема має єдину внутрішню 8-розрядну шину, по якій передаються дані, коди команд та адреси.

Структурна схема містить пристрій керування ПК, дешифратор команд ДШК, регістр команд РК, арифметично-логічний пристрій АЛП, акумулятор А, часовий акумулятор ЧА, часовий регістр ЧР, регістр прапорців *F*, блок 8-розрядних регістрів загального призначення РЗП, мультиплексор, покажчик стеку (*Stack Pointer SP*), покажчик команд (*Instruction Pointer IP*), буферний регістр адреси БА, буферний регістр даних БД, схему інкремента-декремента СІД.

Пристрій керування відповідно до дешифрованих кодів команд та зовнішніх керуючих сигналів генерує керуючі сигнали для всіх блоків структурної схеми.

Дешифратор команд формує сигнали для пристрою керування згідно з дешифрованим кодом команди. У 8-розрядному *регістрі команд* зберігається машинний код команди (один байт).

Арифметико-логічний пристрій — це комбінаційна схема на основі суматора і логічних елементів, який сигналами з виходів пристрою керування налагоджується на ту чи іншу арифметичну або логічну операцію — додавання, віднімання, І, АБО, ВИКЛЮЧАЛЬНЕ АБО, НІ, зсув.

Акумулятор є 8-розрядним регістром, в якому зберігається один з операндів у двоопераидних командах, а також результат операції. Наприклад, у команді додавання

$$ADD\ B;\ A+B \rightarrow A$$

вказано лише один операнд — 8-розрядний регістр *B*. Регістр *B* — один із регістрів загального призначення (РЗП). У де-

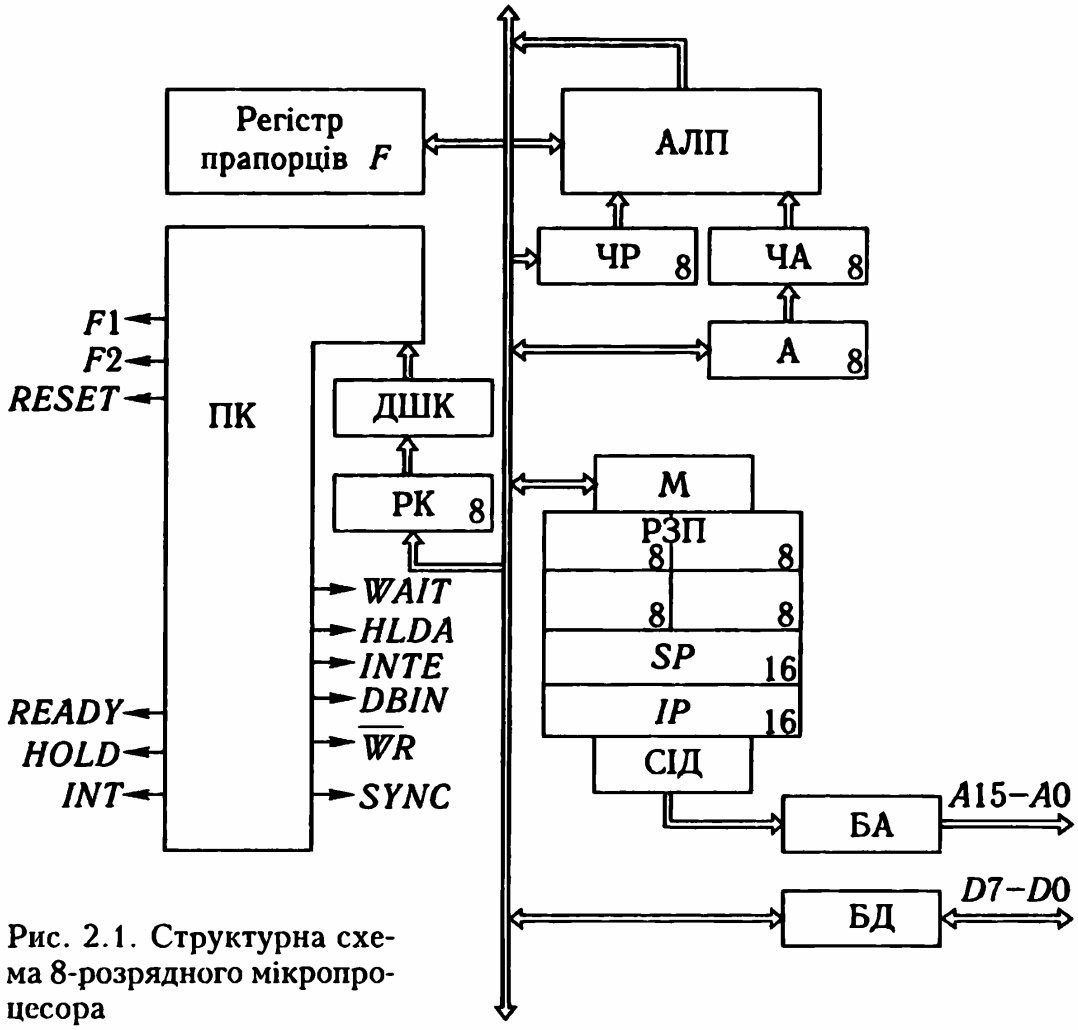


Рис. 2.1. Структурна схема 8-розрядного мікропроцесора

яких МП РЗП позначаються літерами латинського алфавіту: *B*, *C*, *D*, *H*, *L*, в інших — *R*0, *R*1, *R*2,... . Другим операндом є акумулятор. Результат додавання вмісту акумулятора та регістра *B* переноситься в акумулятор, що символічно записується в коментарі до команди.

Часовий акумулятор та ***часовий регістр*** — це 8-розрядні буферні регістри, які дозволять відокремити входи АЛП від його виходу, тобто виключити «гонку» сигналів;

Регістр прапорців *F* (*Flags*-прапорці), або ознак, — кілька тригерів (п'ять або шість), які встановлюються в одиничний (або скидаються в нульовий) стан залежно від результату операції в АЛП;

Регістри загального призначення — блок 8-розрядних РЗП, в яких зберігаються дані та проміжні результати. Цей блок РЗП можна розглядати як швидкодіючий ОЗП, що має найбільшу швидкодію серед ОЗП різних типів, оскільки він розміщений безпосередньо на кристалі ВІС МП. Деякі типи 8-розрядних процесорів, крім 8-розрядних РЗП, містять

16-розрядні індексні регістри для організації непрямої адресації, інші — пропускають звернення до пари 8-розрядних регістрів як до одного 16-розрядного;

Мультиплексор — пристрій, що з'єднує один з регістрів РЗП із внутрішньою шиною МП.

Покажчик стеку SP *(Stack Pointer)* — 16-розрядний регістр, в якому зберігається адреса останньої зайнятої комірки стеку;

Покажчик команд IP *(Instruction Pointer)* — 16-розрядний регістр, в якому зберігається адреса команди, що виконується. Після вибірки з пам'яті програм кожного байта команди вміст *IP* збільшується на одиницю. У літературі цей регістр інколи називають *PC (Program Counter)* — програмний лічильник.

Буферний регістр адреси та ***буферний регістр даних*** — регістри з трьома станами виходу, призначені для формування відповідно сигналів на лініях шин адреси і даних.

Схема інкремента-декремента — пристрій, що дає змогу без участі АЛП збільшити або зменшити на одиницю вміст одного з регістрів РЗП, *IP* або *SP*.

Конструктивно ВІС 8-розрядного процесора виконано в корпусі з 40 виводами, з яких 16 припадає на шину адреси, 8 — на шину даних, 2 (4) — на ввімкнення живлення, а інші — на лінії шини керування. Основні лінії шини керування показано на рис. 2.1:

F1, *F2* — вхід двох послідовностей імпульсів синхронізації, що не перекриваються (рис. 2.2);

RESET — вхід сигналу початкового встановлення (скидання);

READY — вхід сигналу готовності зовнішнього пристрою або пам'яті до обміну; використовується для організації обміну з менш швидкодіючими (порівняно з МП) пристроями;

WAIT — вихід сигналу підтвердження-очікування; активний рівень сигналу свідчить про те, що процесор перейшов у режим очікування і виконує холості такти;

HOLD — вхід сигналу запиту *прямого доступу до пам'яті* або запит захоплення шин; використовується для організації обміну з пристроями, швидкодія яких вища, ніж швидкодія процесора;

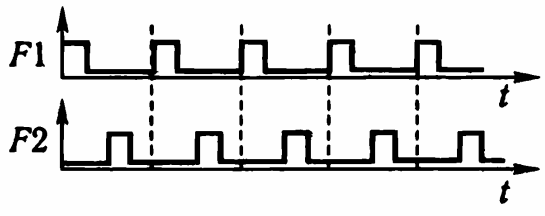


Рис. 2.2. Часові діаграми імпульсів синхронізації *F1* і *F2*

HLDA (HoLD Acknowledge) — вихід сигналу підтвердження прямого доступу до пам'яті; активний рівень цього сигналу свідчить

про те, що процесор перевів свої шини адреси, даних та керування у високоімпедансний стан;

INT (*INTerrupt*) — вхід сигналу запиту переривання;

INTE (*INTerrupt Enable*) — вихід сигналу дозволу переривання;

DBIN (*Data Bus IN*) — вихід сигналу читання; високий рівень (*H*-рівень) цього сигналу вказує, що двонапрямлена шина даних знаходиться у режимі прийому інформації;

$\overline{WR}$ (*WRite*) — вихід сигналу запису; низький рівень цього сигналу свідчить про те, що двонапрямлена шина даних знаходиться у режимі видавання інформації;

SYNC (*SYNChronization*) — вихід сигналу синхронізації; високий рівень цього сигналу означає, що на шині даних передається байт стану, який використовується для формування деяких керуючих сигналів.

Схеми окремих МП відрізняються кількістю та позначенням регістрів, а також деякими керуючими сигналами. Наприклад, у МП *i*8085 замість двох сигналів *F*1 і *F*2 використовується один сигнал синхронізації *CLK* (*CLocK*); замість сигналу *DBIN* — сигнал читання $\overline{RD}$ (*ReaD*). Нульовий рівень цього сигналу свідчить про те, що двонапрямлена шина даних знаходиться у режимі приймання інформації. У МП *i*8085 є додатковий сигнал *M* / $\overline{IO}$ (*Memory/Input-Output*) — ознака звернення до пам'яті (логічна одиниця) або до пристрою введення-виведення (логічний нуль), але немає сигналу *SYNC*.

Схема (див. рис. 2.1) працює так. Під час ввімкнення живлення або під час формування сигналу початкового встановлення *RESET* вміст покажчика команд *IP* набуває нульового значення і починається машинний цикл вибірки команди з пам'яті. Вміст комірки пам'яті за нульовою адресою через буферний регістр даних та внутрішню шину МП надходить у регістр команд, а після цього — у дешифратор команд. Відповідно до дешифрованих кодів команд і зовнішніх сигналів синхронізації та керування пристрій керування формує керуючі імпульси для кожної мікрооперації команди.

Програмною моделлю МП називають сукупність програмно-доступних регістрів, тобто тих регістрів, уміст яких можна зчитати або змінити за допомогою команд. Програмну модель МП складають акумулятор, РЗП, регістр прапорців, покажчик стеку та команд.

Організація пам'яті. Максимально можлива ємність пам'яті з прямою адресацією визначається кількістю розрядів шини адреси. Більшість 8-розрядних процесорів (*i*8080, *i*8085, *Z*80,

Motorola 6800) мають 16-розрядну шину адрес, тобто дають змогу адресувати 2^{16} = 64 Кбайт пам'яті.

Мікропроцесори з 8-розрядною шиною даних мають чотири режими адресації операндів:

• пряма адресація. У цьому режимі другий і третій байти команд містять адресу операнда;

• регістрова адресація. У мнемоніці команди вказується позначення РЗП, в якому знаходиться операнд;

• безпосередня адресація. У цьому режимі в команді вказується 8- або 16-бітовий операнд у другому або у другому та третьому байтах команди. Операнд у цьому випадку знаходиться у пам'яті програм;

• непряма регістрова адресація. У команді вказується регістр (або пара регістрів), який містить адресу комірки пам'яті.

Організація введення-виведення. 8-Розрядні МП дають змогу передати або прийняти дані із зовнішніх ПВВ. Пристрої введення-виведення з'єднані із системною шиною МП системи за допомогою портів введення-виведення, які є 8-розрядними регістрами зі схемами вибірки та керуванням читанням-записом. Кількість таких пристроїв визначається можливим діапазоном 8-розрядних адрес портів, тобто 2^8 = 256 портів введення і 256 портів виведення. Як порти введення можуть бути використані буферні регістри, наприклад КР580ИР82, КР589ИР12 або інтерфейс введення-виведення паралельної інформації КР580ВВ55.

Введення або виведення даних може здійснюватися двома способами: з використанням окремого адресного простору ПВВ або з використанням спільного з пам'яттю адресного простору, тобто з відображенням на пам'ять.

Перший спосіб дає змогу виконувати введення і виведення даних за командами введення *IN* та виведення *OUT*. Використання другого способу передбачає розташування адрес портів у спільному з пам'яттю адресному просторі. При цьому операції звернення до портів не відрізняються від операцій звернення до пам'яті.

Виконання команд в МП *i*8080. Кожна команда в МП виконується впродовж *командного циклу*. Командний цикл складається з *циклу вибірки команди* та *циклу виконання команди* (рис. 2.3).

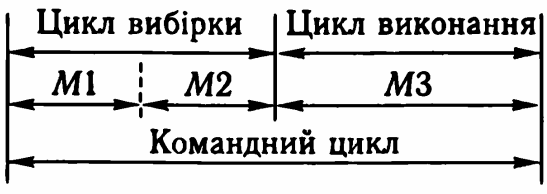


Рис. 2.3. Приклад командного циклу 8-розрядного МП

Тривалість *циклу вибірки команди* залежить від формату команди (кількості байтів у машинному коді команди). Команди займають від

одного до трьох байтів у програмній пам'яті. Багатобайтові команди зберігаються у сусідніх комірках пам'яті. Для вибірки однобайтової команди (наприклад, додавання акумулятора *A* і регістра *B-ADD B*) потрібне одне звернення до пам'яті, для вибірки трибайтової команди (наприклад, виклику підпрограми за адресою *ADDR-CALL ADDR*) — три звернення. Тривалість *циклу виконання команди* залежить від способу адресації операндів. Так, під час виконання команд з регістровою адресацією додаткове звернення до пам'яті для читання операнда не використовується, а в командах з непрямою адресацією таке звернення обов'язкове.

Тому тривалість командного циклу в МП *i*8080 є різною для різних команд і визначається кількістю звернень до пам'яті або до зовнішнього пристрою. Інтервал, упродовж якого здійснюється одне звернення процесора до пам'яті чи до зовнішнього пристрою, визначається як машинний цикл *M*. Отже, командний цикл процесора складається з деякої кількості машинних циклів (залежно від типу команди). У наведеному на рис. 2.2 прикладі цикл вибірки має два машинних цикли (*M*1 і *M*2), а цикл виконання — один машинний цикл (*M*3). У команді може бути від одного (для однобайтових команд з регістровою адресацією) до п'яти (для трибайтових складних команд) машинних циклів.

Машинний цикл, у свою чергу, поділяється на деяку кількість машинних тактів *T*, упродовж кожного з яких виконується елементарна дія (мікрооперація) у процесорі. Кількість тактів у циклі визначається кодом команди і становить від 3 до 5. Тривалість такту задається періодом імпульсів синхронізації і визначається як інтервал часу між фронтами двох сусідніх імпульсів послідовності *F*1, яка формується зовнішніми ланцюгами. Отже, командний цикл МП *i*8080 складається з певної кількості машинних циклів, а кожний машинний цикл — з визначеної кількості тактів, упродовж яких виконуються ті чи інші елементарні дії у процесорі.

Для синхронізації процесора з пам'яттю та зовнішніми пристроями, що характеризуються меншою швидкодією, для організації роботи в режимі прямого доступу до пам'яті (ПДП) та зупинки процесора передбачено три особливих режими: *очікування*, *захоплення шин*, *зупинки*, тривалість яких має довільну, але завжди кратну тривалості такту *T* величину.

Залежно від дій, що виконує МП, розрізняють такі типи машинних циклів:

ВИБІРКА (читання першого байта команди);

ЧИТАННЯ ПАМ'ЯТІ (читання другого та третього байтів команди, читання операнда);

ЗАПИС У ПАМ'ЯТЬ;
ЧИТАННЯ СТЕКУ;
ЗАПИС У СТЕК;
ВВЕДЕННЯ даних із зовнішнього пристрою;
ВИВЕДЕННЯ даних на зовнішній пристрій;
ПЕРЕРИВАННЯ;
ЗУПИНКА;
ПЕРЕРИВАННЯ ПІД ЧАС ЗУПИНОК.

Першим машинним циклом команди завжди є цикл ВИБІРКА, впродовж якого здійснюється вибірка з пам'яті байта коду команди за адресою, що визначається вмістом покажчика команд. Вміст покажчика у циклі збільшується на одиницю. Однобайтові команди з регістровою адресацією потребують для виконання лише одного циклу ВИБІРКА.

Для вибірки дво- або трибайтових команд крім циклу ВИБІРКА потрібні ще один або два машинні цикли для читання другого або другого і третього байтів команди. Ці цикли визначаються як цикли ЧИТАННЯ ПАМ'ЯТІ. Цикл ЧИТАННЯ ПАМ'ЯТІ необхідний також для вибірки операнда під час виконання команд з непрямою або прямою адресацією. Для запису операндів або зберігання адрес під час виконання відповідних команд пересилання потрібні машинні цикли ЗАПИС У ПАМ'ЯТЬ.

У командах зі зверненням до стеку виконуються цикли ЧИТАННЯ СТЕКУ та ЗАПИС У СТЕК. Адреси пам'яті визначаються покажчиком стеку *SP*. Для виконання команд введення-виведення виконуються машинні цикли ВВЕДЕННЯ та ВИВЕДЕННЯ; для організації переривань програми та зупинки процесора — цикли ПЕРЕРИВАННЯ, ЗУПИНКА, ПЕРЕРИВАННЯ ПІД ЧАС ЗУПИНКИ.

Кожний машинний цикл процесора ідентифікується байтом, який називають *байтом стану*. Байт стану видається на шину даних на початку кожного машинного циклу і супроводжується одночасним видаванням сигналу *SYNC* на однойменний контакт ВІС МП. Байт стану несе інформацію про наступні дії процесора. Його можна запам'ятати за сигналом *SYNC* і сформувати такі керуючі сигнали, які не вдалося вивести в явному вигляді на контакти ВІС МП (через обмежену кількість контактів мікросхеми). Байти стану для процесора *i*8080 наведено в табл. 2.1.

Часову діаграму машинного циклу ВИБІРКА (ЧИТАННЯ ПАМ'ЯТІ) для процесора *i*8080 показано на рис. 2.4, а циклу ЗАПИС У ПАМ'ЯТЬ — на рис. 2.5. Такти відлічуються за передніми фронтами послідовності *F*1, а мікрооперації в кожному такті визначаються переднім фронтом сиг-

Таблиця 2.1. **Байти стану для різних типів машинних циклів**

Розряд шини даних	Тип машинного циклу									
	ВИБІРКА	ЧИТАННЯ ПАМ'ЯТІ	ЗАПИС У ПАМ'ЯТЬ	ЧИТАННЯ СТЕКУ	ЗАПИС У СТЕК	ВВЕДЕННЯ	ВИВЕДЕННЯ	ПЕРЕРИВАННЯ	ЗУПИНКА	ПЕРЕРИВАННЯ ПІД ЧАС ЗУПИНКИ
*D*0	0	0	0	0	0	0	0	1	0	1
*D*1	1	1	0	1	0	1	0	1	1	1
*D*2	0	0	0	1	1	0	0	0	0	0
*D*3	0	0	0	0	0	0	0	0	1	1
*D*4	0	0	0	0	0	0	1	0	0	0
*D*5	1	0	0	0	0	0	0	1	0	1
*D*6	0	0	0	0	0	1	0	0	0	0
*D*7	1	1	0	1	0	0	0	0	1	0

палу *F*2. За тактової частоти 2 МГц тривалість такту становить 0,5 мкс. Сигнали на лініях шин $A15-A0$ ($D7-D0$) зображено на рис. 2.4 на одній часовій діаграмі у вигляді ліній *L*- та *H*-рівнів одночасно. На діаграмі вказано характер інформації, наявної на шині. Пунктирною лінією позначено високоімпедансний стан ліній шин (*z*-стан).

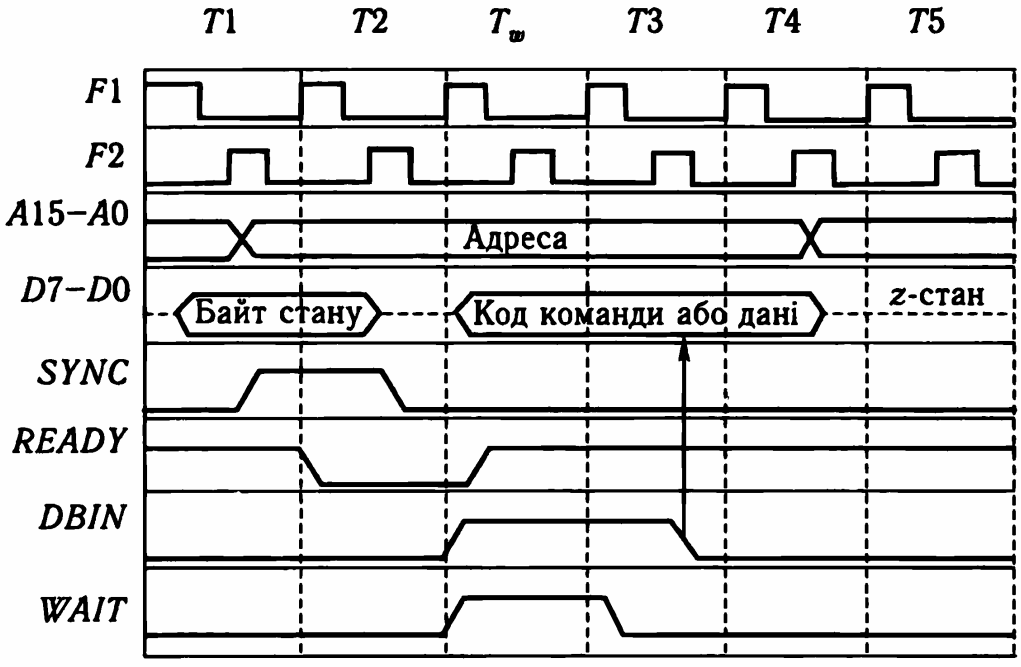


Рис. 2.4. Цикл ВИБІРКА І ЧИТАННЯ ПАМ'ЯТІ

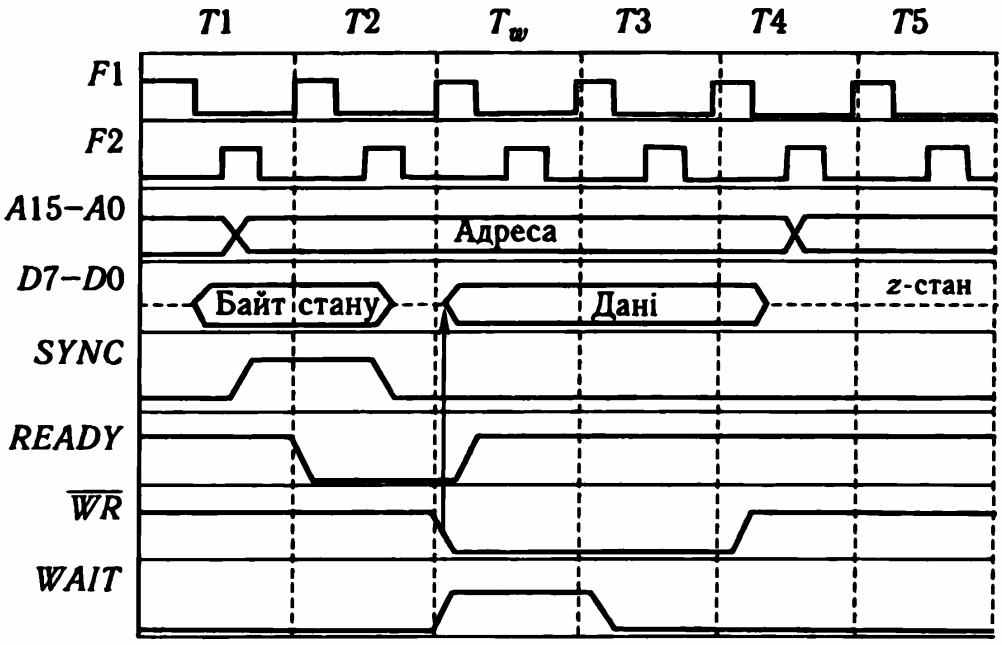


Рис. 2.5. Цикл ЗАПИС У ПАМ'ЯТЬ

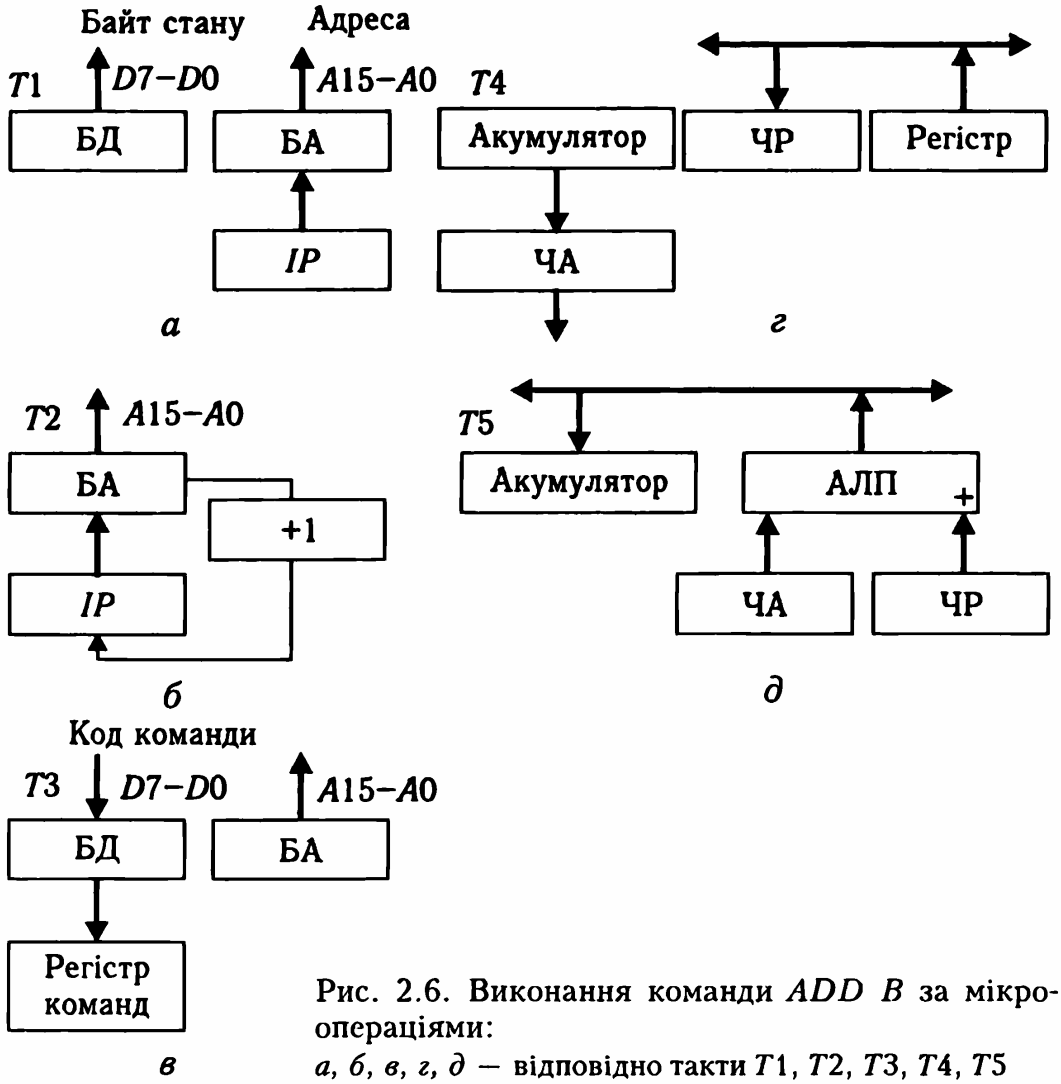


Рис. 2.6. Виконання команди *ADD B* за мікроопераціями:
а, б, в, г, д — відповідно такти *T*1, *T*2, *T*3, *T*4, *T*5

У першому машинному такті $T1$ на шину адреси (лінії $A15-A0$) видається адреса — вміст вказівника команд IP, якщо виконується цикл ВИБІРКА або вміст покажчика адреси — якщо виконується цикл ЧИТАННЯ ПАМ'ЯТІ. Водночас на шину даних (лінії $D7-D0$) видається байт стану і формується сигнал $SYNC$ на однойменному виводі ВІС МП.

У другому такті $T2$ закінчується надходження байта стану і сигналу $SYNC$, тривалість яких дорівнює одному такту. У машинному циклі ВИБІРКА вміст IP збільшується для адресації наступного байта команди або наступної команди. У цьому самому такті пристрій керування МП здійснює аналіз сигналів на входах $READY$ і $HOLD$ та контроль виконання команди зупинки HLT. Якщо пам'ять або зовнішній пристрій не готові до обміну ($READY = 0$), оскільки надійшов запит ПДП ($HOLD = 1$) або виконується команда зупинки HLT, то обмін даними не відбувається, і процесор переходить в один з режимів — *очікування, захоплення шин* або *зупинки*. У цих режимах здійснюється очікування сигналу впродовж кількох *тактів очікування* T_w, кількість яких визначається зовнішніми сигналами. На рис. 2.4 у такті $T2$ сигнал $READY$ дорівнює логічному нулю, а у такті T_w — логічній одиниці.

У такті $T3$ залежно від типу машинного циклу здійснюється звернення до пам'яті, стеку або зовнішнього пристрою. Тому в МП вводиться (див. рис. 2.4) або з нього виводиться (див. рис. 2.5) байт команди, адреси або даних. Залежно від типу команди машинний цикл може містити такти $T4$ і $T5$, наприклад, якщо для виконання команди потрібна обробка операндів. В останньому такті команди ($T3$, $T4$ або $T5$) аналізується наявність сигналу запиту переривання INT. Якщо переривання дозволено, то процесор переходить до машинного циклу ПЕРЕРИВАННЯ.

Як приклад розглянемо виконання команди $ADD\ B$ за мікрооперaціями. Команда містить один машинний цикл ВИБІРКА, що виконується за чотири такти, а потрібна для виконання команди мікрооперація п'ятого такту виконується в такті $T2$ наступної команди. На рис 2.6, $а-д$ показано дії МП у кожному такті машинного циклу. У такті $T1$ (рис. 2.6, *а*) на шину адреси видається вміст покажчика команд IP, який в такті $T2$ (рис. 2.6, *б*) збільшується на одиницю для адресації наступної команди. Команда вибирається з пам'яті у такті $T3$ (рис. 2.6, *в*). У такті $T4$ (рис. 2.6, *г*) здійснюється підготовка операндів до додавання: вміст регістра B по внутрішній шині пересилається в часовий регістр, а вміст акумулятора A — в регістр у часовий акумулятор.

У п'ятому такті, який суміщений з тактом *T*2 наступної команди для збільшення швидкодії, виконується додавання операндів. Результат додавання запам'ятовується в акумуляторі.

Особливі режими роботи МП *i*8080. Мікропроцесор *i*8080 має такі особливі режими роботи: переривання, очікування, захоплення шин за прямого доступу до пам'яті, що ініціюються зовнішніми сигналами керування, зупинка (перехід до цього режиму здійснюється програмно).

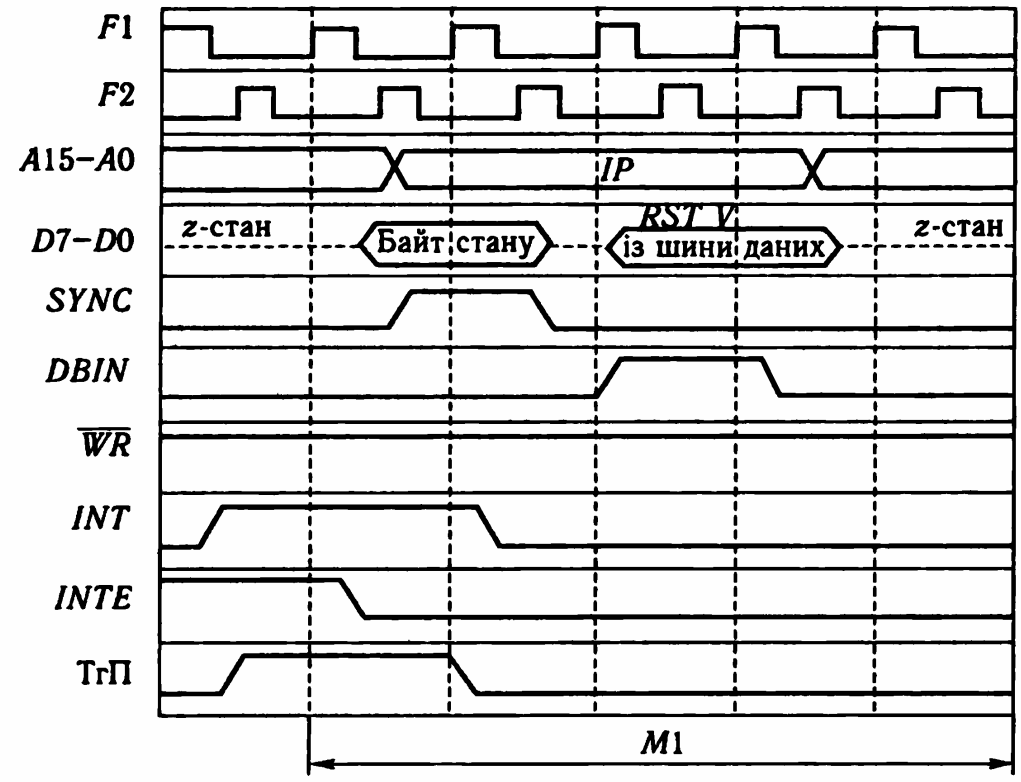
Переривання. У мікропроцесорі *i*8080 є засоби обробки запитів переривань восьми рівнів. Якщо один із зовнішніх пристроїв, з'єднаних із системою переривання МП, ініціює запит переривання (див. рис. 1.4), то система формує сигнал на виводі *INT* МП у вигляді сигналу високого рівня. Одночасно на шину даних система переривання посилає код команди *RST V* (переривання за вектором *V*). Вектор *V* — код, який вказує на адресу початкової команди підпрограми обслуговування цього запиту переривання

Послідовність дій МП у *режимі переривання*:

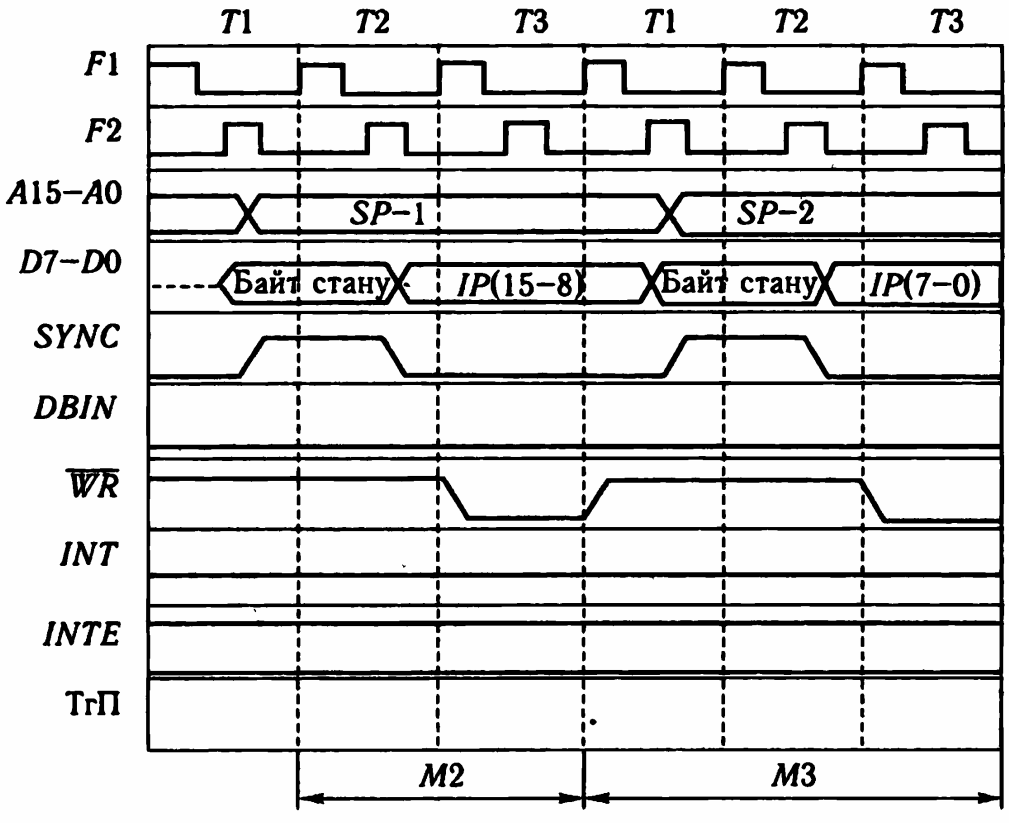
- приймання запиту переривання та блокування входу запиту переривання;
- приймання команди *RST V*;
- зберігання адреси повернення (вмісту покажчика команд) у стеку;
- формування адреси підпрограми обслуговування джерела запиту.

Запити переривання МП *i*8080 приймаються із входу *INT* МП *тригером переривань*, яким керує *тригер дозволу переривань*. У свою чергу, тригер дозволу переривань можна програмним способом установити в стан логічної одиниці або логічного нуля. Стан логічної одиниці тригера дозволу переривань дає змогу приймати переривання із входу *INT*, а стан логічного нуля — це забороняє.

Система переривання може встановити активний рівень сигналу на лінії *INT* у будь-який момент виконання програми, однак приймання його синхронізується так. За наявності сигналу дозволу переривання (тригер дозволу, що знаходиться у стані логічної одиниці) тригер переривань установлюється в останньому такті останнього машинного циклу команди, впродовж виконання якої надійшов запит. Це дає змогу процесору завершити виконання команди перед тим, як почнеться обробка переривання. Якщо переривання не дозволено, тобто тригер дозволу переривань скинуто до нуля, запит переривань із входу *INT* ігнорується. Часову діаграму роботи процесора показано на рис. 2.7, *а, б*.



а



б

Рис. 2.7. Часові діаграми циклу ПЕРЕРИВАННЯ:
а — машинний цикл *M*1; *б* — машинні цикли *M*2 і *M*3; ТгП — тригер переривань

Після прийняття сигналу запиту переривання процесор переходить до виконання циклу ПЕРЕРИВАННЯ, що складається з трьох машинних циклів.

Перший з них *М*1 (рис. 2.7, *а*) призначений для приймання команди *RST V*, а два інших *М*2 і *М*3 — для зберігання адреси повернення в стек (рис. 2.7, *б*). У циклі *М*1 у першому такті *Т*1 у байті стану формується сигнал підтвердження переривання, який використовується для керування читанням команди *RST V*. У такті *Т*3 процесор приймає по шині даних байт команди *RST V*, що формується системою переривання. У тактах *Т*4, *Т*5 циклу ПЕРЕРИВАННЯ здійснюється формування адреси першої комірки стеку, яка відведена для зберігання адреси повернення з підпрограми обслуговування запиту переривання. У циклах *М*2 та *М*3 здійснюється завантаження адреси повернення (вмісту покажчика команд) у стек. У наступному циклі формується перша команда підпрограми обслуговування переривання за адресою, вказаною в команді *RST V*. Наступне керування МП покладається на підпрограму.

В окремому випадку за допомогою підпрограми здійснюється зберігання вмісту основних робочих регістрів процесора, керування тригером дозволу переривання, відтворення вмісту основних робочих регістрів та повернення до основної програми (відновлення вмісту покажчика команд).

Захоплення шин. *Режим захоплення шин* використовується для організації виконання операцій прямого доступу до пам'яті. Для цього процесор має вхідний вивід *HOLD* запиту захоплення шин та вихідний вивід *HLDA* підтвердження захоплення. Зовнішній пристрій запитує режим прямого доступу до пам'яті сигналом високого рівня на лінії *HOLD*. При цьому процесор зупиняє виконання операцій і від'єднується від зовнішніх шин даних та адреси. Лінії шин переходять у високоімпедансний стан.

Процесор підтверджує прийняття запиту прямого доступу до пам'яті встановленням високого потенціалу на виході підтвердження захоплення *HLDA*. Поки діє сигнал на вході *HOLD*, шини процесора знаходяться в розпорядженні зовнішнього пристрою, який надіслав запит прямого доступу до пам'яті. Сигнали керування обміном інформацією між зовнішнім пристроєм та пам'яттю формуються спеціальною ВІС — контролером прямого доступу до пам'яті.

Часову діаграму роботи процесора у режимі захоплення шин у циклі ЧИТАННЯ ПАМ'ЯТІ зображено на рис. 2.8. Сигнал *HOLD* сприймається процесором у такті *Т*2. За наявності сигналу готовності зовнішнього пристрою на вході

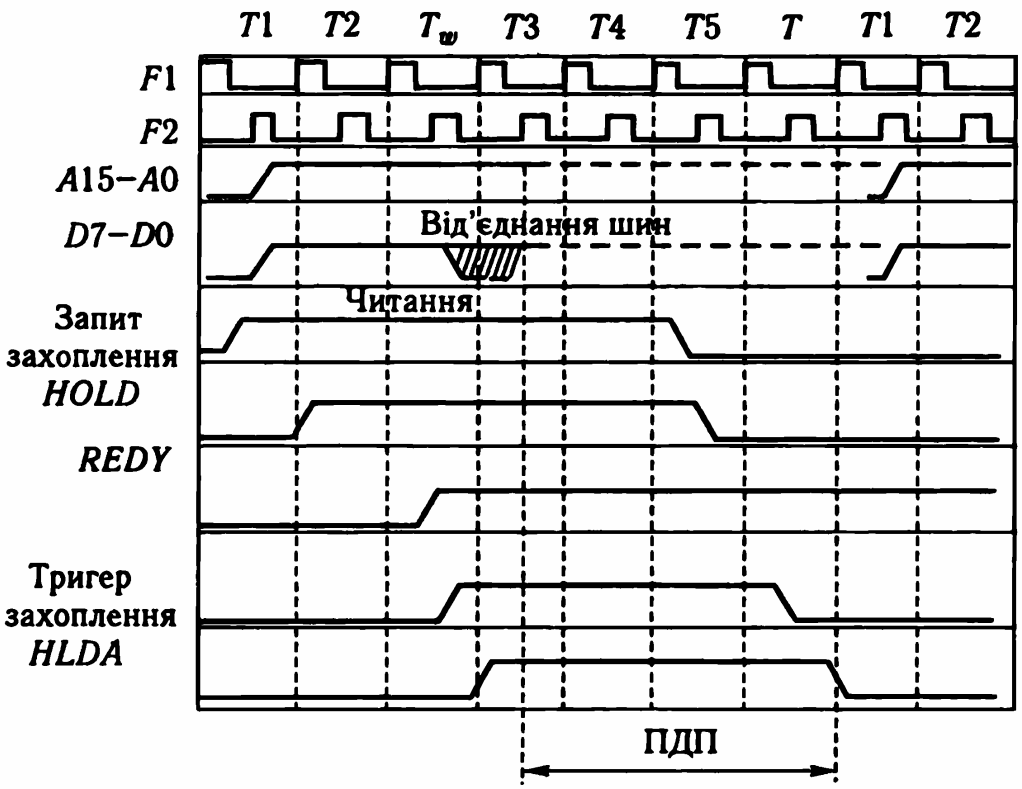


Рис. 2.8. Цикл ЧИТАННЯ ПАМ'ЯТІ в режимі ПДП

READY встановлюється високий рівень на вході внутрішнього тригера захоплення, завдяки чому по фронту наступного імпульсу *F*1 вихідний сигнал МП *HLDA* перемикається у стан логічної одиниці. У процесі виконання циклів читання або введення процесор підтверджує захоплення на початку такту *T*3 після закінчення читання. У циклах записування та виведення це здійснюється у такті *T*4 після закінчення запису. В обох випадках шини процесора переводяться у високоімпедансний стан по фронту імпульсу *F*2, наступного за імпульсом *F*1, після якого виконувалося перемикання виходу *HLDA*.

Процесор з режиму захоплення виходить так. Після закінчення асинхронного сигналу запиту захоплення на вході *HOLD* імпульсом *F*2 тригер переходить у стан логічного нуля на передньому фронті імпульсу *F*1 на виході *HLDA* підтвердження захоплення формується сигнал низького потенціалу. Процесор переходить до виконання наступного машинного циклу.

Зупинка. Процесор входить у режим зупинки після виконання команди зупинки *HLT*. Виконання дій МП проілюстровано часовими діаграмами (рис. 2.9). Перехід у режим зупинки виконується за два машинних цикли. У першому машинному циклі ВИБІРКА здійснюється зчитування з па-

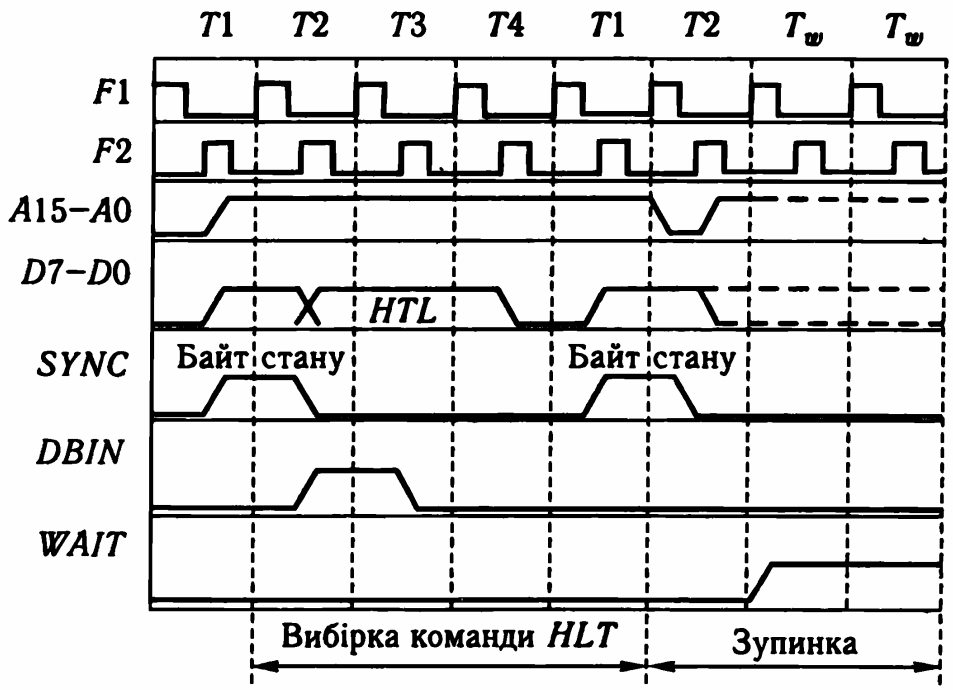


Рис. 2.9. Цикл ЗУПИНКА

м'яті першого байта команди *HLT*. У другому машинному циклі ЗУПИНКА після закінчення такту *T*2 процесор переходить у режим зупинки, за якого шини даних і адреси переходять у високоімпедансний стан, а процесор виконує такти очікування T_w. Режим зупинки підтверджується бітом *D*3 у байті стану, який видається у такті *T*2 (див. табл. 2.1).

Вихід з режиму зупинки можна здійснити трьома способами:

- поданням сигналу на лінію *RESET* (при цьому покажчик команд набуває нульового значення і процесор переходить до машинного циклу ВИБІРКА команди за першою адресою);
- поданням сигналу на вхід *HOLD*, завдяки чому процесор переходить до виконання циклу ЗАХОПЛЕННЯ. Після закінчення цього сигналу процесор входить у режим зупинки за переднім фронтом імпульсу *F*1;
- поданням сигналу переривання (за наявності сигналу дозволу переривань на виході *INTE*), завдяки чому процесор по фронту імпульсу *F*1 переходить у режим *T*1 машинного циклу ПЕРЕРИВАННЯ. Для реалізації цієї можливості треба перед початком режиму зупинки забезпечити встановлення тригера дозволу переривань після виконання команди дозволу переривання *EI*.

Оброблення запитів захоплення шин і переривання під час зупинки зображено на рис. 2.10. Нехай проце-

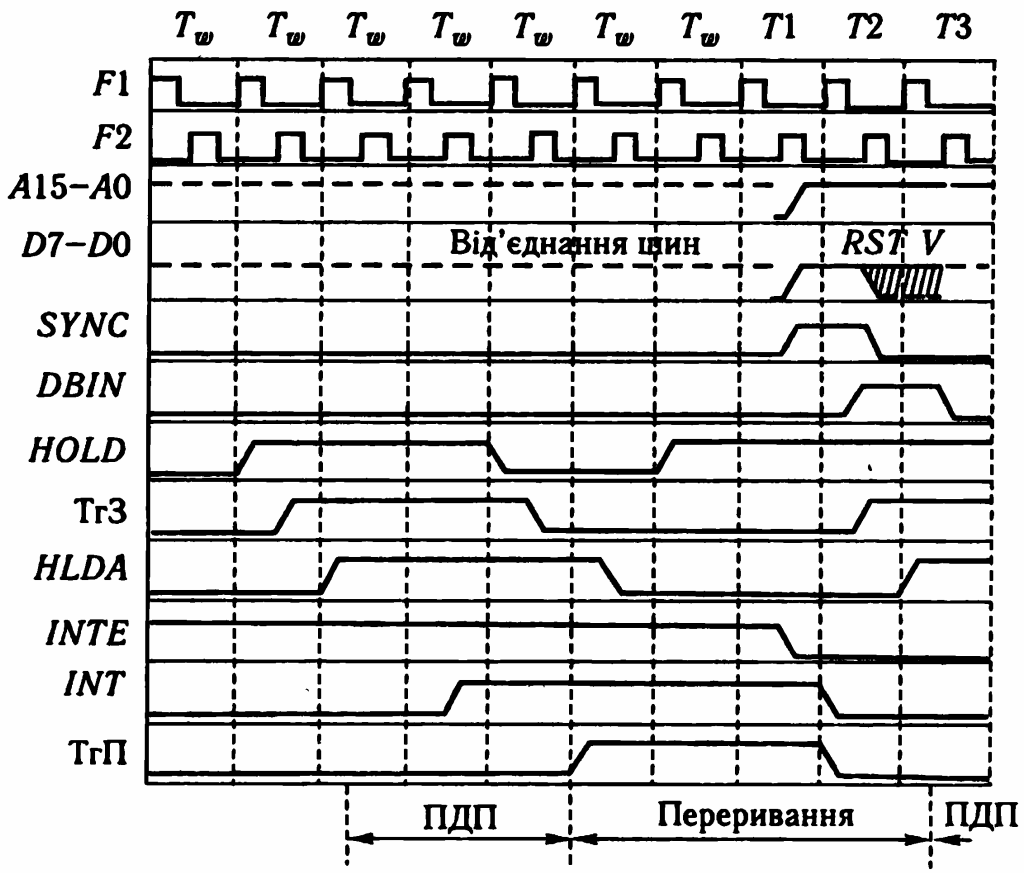


Рис. 2.10. Цикли ЗАХОПЛЕННЯ шин і ПЕРЕРИВАННЯ під час ЗУПИНКИ

сор знаходиться у режимі ЗУПИНКА, тоді лінії шин адрес та даних знаходяться у високоімпедансному стані і процесор виконує такти T_w.

З надходженням запиту захоплення шин *HOLD* у наступному такті встановлюється тригер захоплення. Потім процесор видає сигнал підтвердження захоплення *HLDA* і здійснює прямий доступ до пам'яті. Якщо виконується послідовність дій у режимі ПДП, то запити переривання не сприймаються процесором до закінчення обміну, а якщо переривання дозволено і надійшов запит переривання *INT*, то після закінчення режиму ПДП МП переходить до машинного циклу ПЕРЕРИВАННЯ ПІД ЧАС ЗУПИНКИ. Після надходження у цьому циклі запиту захоплення *HOLD* він ігнорується МП до завершення виконання циклу читання команди *RST*. Потім МП переходить до режиму ПДП. Послідовність дій МП у режимі переривання завершується після закінчення режиму ПДП.

Увімкнення мікропроцесора. З поданням напруги живлення процесор починає функціонувати. Напругу живлення МП треба вмикати разом з поданням сигналу на лінію *RESET*.

Тривалість цього сигналу має становити не менш як три періоди імпульсів синхронізації. За сигналом *RESET* вміст покажчика команд набуває нульового значення, і процесор починає виконувати дії, що відповідають машинному циклу ВИБІРКА. У результаті МП починає виконувати команду, код якої розміщений у нульовій комірці пам'яті. Отже, запуск програми починається за сигналом на лінії *RESET*. Першою командою програми має бути команда безумовного переходу *JMP ADR*, що здійснює перехід до команди, яка знаходиться у довільному місці пам'яті.

Уміст регістрів загального призначення і регістра прапорців залишається невизначеним, поки його не встановлять відповідні команди програми.

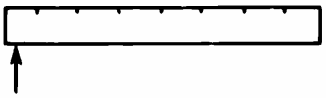
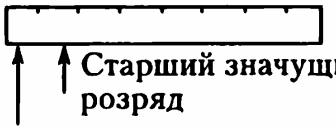
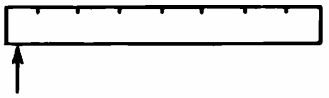
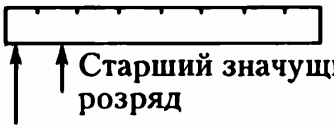
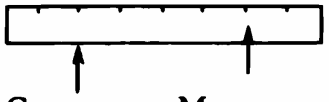
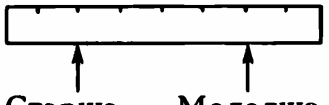
2.2. Однокристальні 16-розрядні мікропроцесори

До 16-розрядних МП першого покоління належать МП *i*8086/*i*8088 та *i*80186/*i*80188, до МП другого — *i*80286. Велика інтегральна схема МП *i*8086 з геометричними розмірами 5,5 × 5,5 мм має 40 контактів, містить близько 29 000 транзисторів і споживає 1,7 Вт від джерела живлення +5 В, тактова частота — 5,8 або 10 МГц.

Мікропроцесор виконує операції з 8- та 16-розрядними даними, наведеними у двійковому або двійково-десятковому вигляді, може обробляти певні біти та рядки або масиви даних. Він має вбудовані апаратні засоби множення і ділення. Формати даних і виконувані операції наведено в табл. 2.2 Мікропроцесор має внутрішній надоперативний запам'ятовувальний пристрій (НОЗП) ємністю 14 × 16 байт. Шина адреси 20-розрядна, що дає змогу безпосередньо адресувати до 2^{20} = 1 048 576 комірок пам'яті (1 Мбайт). Простір адрес введення-виведення становить 64 Кбайт. У ВІС *i*8086 реалізовано багаторівневу векторну систему переривань з кількістю векторів до 256. Передбачено також організацію прямого доступу до пам'яті, після чого МП припиняє роботу та переводить у третій стан шини адреси, даних і керування.

Середня тривалість виконання команди займає 12 тактів. Особливістю МП *i*8086 є можливість часткової реконфігурації апаратної частини для забезпечення роботи у двох режимах — мінімальному і максимальному. Режими роботи задаються апаратно. У *мінімальному режимі*, що використовується для побудови однопроцесорних систем, МП самостійно формує всі сигнали керування внутрішнім системним інтер-

Таблиця 2.2. **Формати даних і операцій, що виконуються МП *i*8086**

Тип даних	Формат	Діапазон	Операції
Байт без знака	7 6 5 4 3 2 1 0  Старший значущий розряд	0...255	Додавання, віднімання, множення, ділення
Байт зі знаком	7 6 5 4 3 2 1 0  Старший значущий розряд Знаковий розряд	−128...+127	Те саме
Слово без знака	15 14 ..... 1 0  Старший значущий розряд	0...65 535	− « −
Слово зі знаком	15 14 ..... 1 0  Старший значущий розряд Знаковий розряд	−32 768 ... +32 767	− « −
Упаковане двійково-десяткове число	7 6 5 4 3 2 1 0  Старша тетрада (0−9) Молодша тетрада (0−9)	0...99	Додавання, віднімання з корекцією
Розпаковане двійково-десяткове число	7 6 5 4 3 2 1 0  Старша тетрада (0) Молодша тетрада (0−9)	0...9	Додавання, віднімання, множення, ділення з корекцією

Примітка. Знакові числа подають у додатковому коді.

фейсом. У *максимальному режимі*, який використовується для побудови мультипроцесорних систем, МП формує на лініях стану двійковий код, який залежить від типу циклу шини. Відповідно до цього коду системний контролер К1810ВГ88 формує сигнали керування шиною. Контакти, які звільнилися після кодування інформації, використовуються для керування мультипроцесорним режимом. Під час використання арифметичного співпроцесора слід обирати максимальний режим.

Структурна схема. У МП *i*8086 застосовано конвеєрну архітектуру, що дає змогу суміщувати у часі цикли вибирання команди та вибірки з пам'яті кодів наступних команд. Це досягається паралельною роботою двох порівняно незалежних пристроїв — операційного пристрою та шинного інтерфейсу. Структурну схему МП *i*8086 зображено на рис. 2.11. Операційний пристрій виконує команду, а шинний інтерфейс здійснює взаємодію із зовнішньою шиною — виставляє адреси, зчитує коди команд, операнди, записує результати обчислень у пам'ять або пристрої введення-виведення.

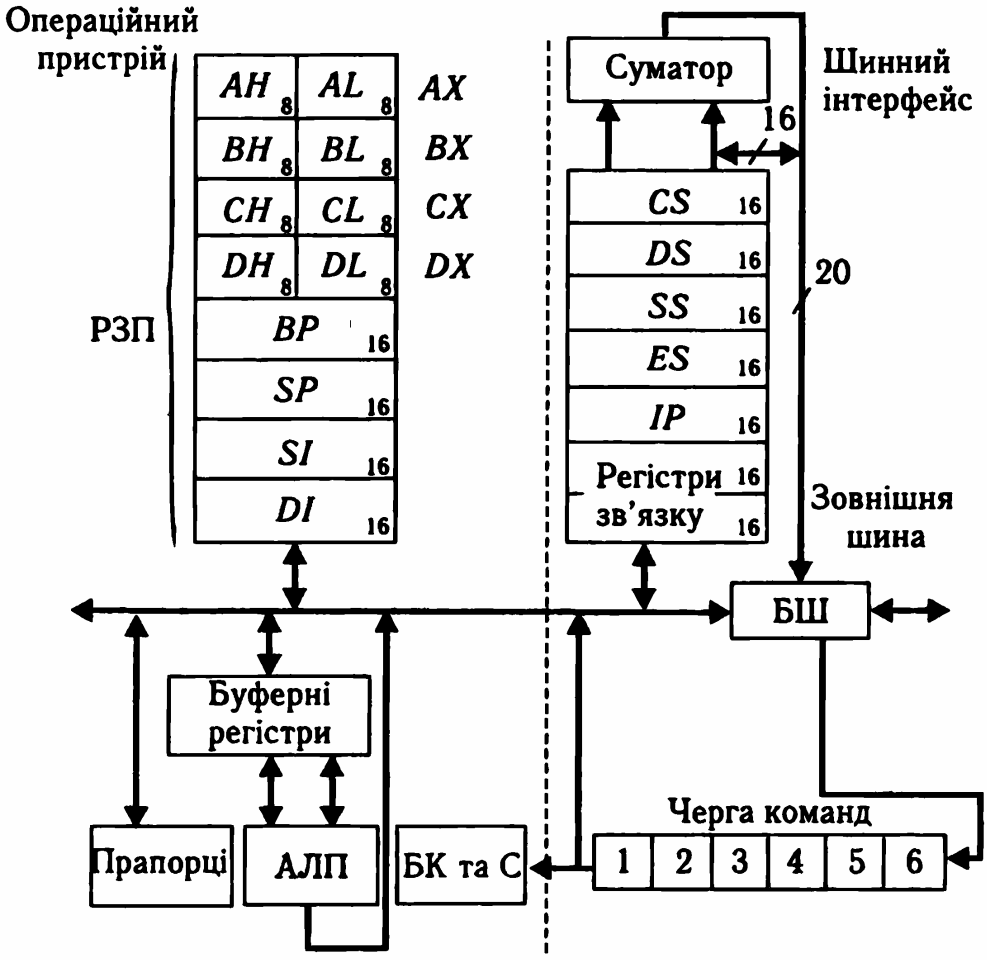


Рис. 2.11. Структурна схема мікропроцесора *i*8086

Операційний пристрій складається з РЗП, призначених для зберігання проміжних результатів — даних та адрес; АЛП з буферними регістрами; регістра прапорців; блоку керування та синхронізації, який дешифрує коди команд і генерує керуючі сигнали для всіх блоків схеми МП. Шинний інтерфейс складається з шестибайтової регістрової пам'яті, яку називають *чергою команд*, чотирьох сегментних регістрів: *CS*, *DS*, *ES*, *SS*, покажчика команд *IP*, суматора допоміжних регістрів зв'язку і буферних схем шин адреси-даних. Черга команд працює за принципом *FIFO* (*First Input — First Output*, тобто *перший прийшов — перший пішов*) і зберігає на виході порядок надходження команд. Довжина черги — 6 байт. Якщо операційний пристрій зайнятий виконанням команди, шинний інтерфейс самостійно ініціює випереджальну вибірку кодів команд з пам'яті у чергу команд. Вибирання з пам'яті чергового командного слова здійснюється тоді, коли в черзі виявляється два вільних байти. Черга збільшує швидкодію процесора у разі послідовного виконання команд. Під час вибирання команд переходів, викликів і повернень з підпрограм та оброблення запитів переривань черга команд скидається і вибирання починається з нового місця програмної пам'яті.

Крім того, одним із завдань шинного інтерфейсу є формування фізичної 20-розрядної адреси із двох 16-розрядних слів. Першим словом є вміст одного з сегментних регістрів *CS*, *SS*, *DS*, *ES*, друге слово залежить від типу адресації операнда або коду команди. Складання 16-розрядних слів відбувається зі зміщенням на чотири розряди і здійснюється за допомогою суматора, що входить до складу шинного інтерфейсу.

Призначення контактів. Умовне графічне зображення МП *i*8086 наведено на рис. 2.12. Призначення контактів ВІС залежить від режиму роботи МП. Вісім контактів має подвійне позначення, причому позначення в дужках відповідають максимальному режиму.

У табл. 2.3 наведено призначення контактів МП, однакових для обох режимів, у табл. 2.4 — призначення контактів, що використовуються лише в мінімальному режимі, а в табл. 2.5 — призначення контактів, які використовуються лише в максимальному режимі. Літерою *z* позначено тристабільні виходи, що переводяться у третій високоімпедансний стан під час переходу МП у *режим захоплення* (у дужках наведено альтернативні позначення контактів, що трапляються у літературі).

Лінії стану. Лінії *ST*2—*ST*0 — виходи сигналів стану — ідентифікують тип циклу шини, що виконується згідно з

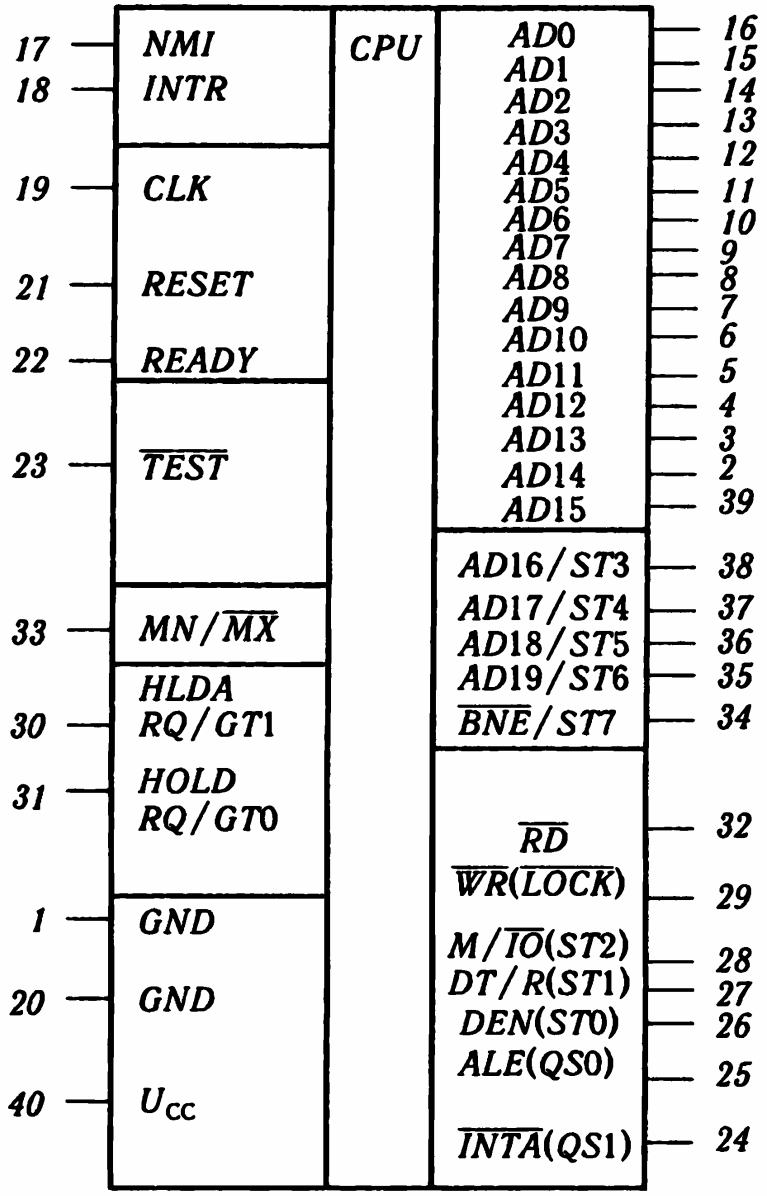


Рис. 2.12. Графічне зображення ВІС МП *i*8086

табл. 2.6. *Циклом шини* називають звернення до комірки пам'яті або зовнішнього пристрою. Це визначення збігається з визначенням машинного циклу для 8-розрядних процесорів (див. розд. 2.1). Однак у 16-розрядних процесорах цикл шини може ініціювати не лише МП, а й арифметичний співпроцесор або спеціалізований процесор введення-виведення.

Початок циклу визначається переходом ліній стану $ST2-ST0$ з пасивного (111) в активний стан, а кінець — зворотним переходом у пасивний стан. Сигнали $ST2-ST0$ подаються на входи контролера шини *i*8288, що дешифрує їх і формує сигнали керування системною шиною $\overline{IOR}$, $\overline{IOW}$, $\overline{MEMR}$, $\overline{MEMW}$, $\overline{INTA}$, ALE, $\overline{DEN}$.

Таблиця 2.3. **Призначення контактів МП *i*8086 для мінімального і максимального режимів**

Позначення	Призначення	Тип
*AD*15—*AD*0	*Address-data* — мультиплексна двонапрямлена шина адреси-даних (*ADB — Address Data Bus*), за якою з розподілом у часі передаються адреси і дані. Адреси передаються в першому такті циклу шини і супроводжуються сигналом *ALE*, а дані — у другій половині циклу шини і супроводжуються сигналом *DEN*	Вхід/вихід (*z*)
$\overline{BHE}$ / *ST*7	*Byte High Enable/Status* 7 — вихідний сигнал дозволу старшого байта-сигналу стану. У першому такті циклу водночас з адресною інформацією передається сигнал $\overline{BHE}$. Активний (нульовий) рівень $\overline{BHE}$ означає, що по старшій половині *AD*15—*AD*8 шини адреси-даних передаються 8-розрядні дані. Сигнал $\overline{BHE}$ використовується для дозволу доступу до старшого банку пам'яті або до зовнішнього пристрою з байтовою організацією, підключеного до старшої половини шини даних. В інших тактах формується сигнал стану *ST*7	Вихід (*z*)
$\overline{RD}$	*Read* — вихідний сигнал читання. Вказує на те, що МП виконує цикл читання	Вихід (*z*)
READY	*Ready* — вхідний сигналу готовності, який підтверджує, що комірка пам'яті або пристрій введення-виведення, який адресується у команді, готовий до взаємодії з МП під час передавання даних	Вхід
INTR	*Interrupt Request* — вхідний сигнал запиту (за одиничного рівня) маскованого переривання. Якщо переривання дозволено, МП переходить до підпрограми обробки переривання. В іншому випадку МП ігнорує цей сигнал	Вхід
$\overline{TEST}$	*Test* — вхідний сигнал перевірки. Сигнал використовується разом з командою очікування *WAIT*, виконуючи яку МП перевіряє рівень сигналу $\overline{TEST}$. Якщо $\overline{TEST}$ = 0, МП переходить до виконання наступної після *WAIT* команди.	Вхід

Продовж. табл. 2.3

Позначення	Призначення	Тип
	Якщо $\overline{TEST}$ = 1, МП знаходиться у стані очікування, виконує холості такти і періодично, з інтервалом $5T_{CLK}$, перевіряє значення сигналу	
CLK, (*CLC*)	*Clock* — вхідні тактові імпульси, які забезпечують синхронізацію роботи МП	Вхід
RESET (*CLR*)	*Reset* (*Clear*) — сигнал апаратного скидання (стан «1»). Переводить МП у початковий стан, за якого скинуті сегментні регістри (крім *CS*, усі розряди якого встановлюються в стан «1»), покажчик команд *IP*, усі прапорці, регістри черги команд і всі внутрішні тригери у пристрої керування. Сигнал *RESET* не впливає на стан загальних регістрів. Під час дії сигналу *RESET* усі виходи, що мають три стани, переводяться у третій стан; виходи, що мають два стани, стають пасивними. Мінімальна тривалість сигналу *RESET* після першого ввімкнення МП становить 50 мкс, а за повторного запуску — чотири такти синхронізації, тобто 0,8 мкс за тактової частоти 5 МГц. Після закінчення сигналу *RESET* починається цикл вибірки команди з пам'яті з адресою 0*FFFFH*: 0000	Вхід
MN / $\overline{MX}$	*Minimum-maximum* — вхід сигналу вибору мінімального або максимального режимів. Сигнал на цьому вході визначає режим роботи МП: 1 — мінімальний, 0 — максимальний	Вхід

Таблиця 2.4. **Призначення контактів МП *i*8086 у мінімальному режимі**

Позначення	Призначення	Тип
$\overline{INTA}$	*Interrupt Acknowledge* — вихідний сигнал підтвердження переривання, що визначає читання вектора переривання	Вихід
ALE	*Address Latch Enable* — вихідний сигнал дозволу фіксації адреси; видається на початку кожного циклу шини і використовується для запису адреси в регістр-фіксатор	Вихід

Продовж. табл. 2.4

Позначення	Призначення	Тип
$\overline{DEN}$ ($\overline{DE}$)	*Data Enable* — вихідний сигнал дозволу даних, що визначає появу даних на шині адреси-даних	Вихід (*z*)
$DT/\overline{R}$ ($OP/\overline{IP}$)	*Data Transmit/Receive* (*Output-Input*) — вихідний сигнал передавання-приймання даних; визначає напрям передавання даних по *ADB*. Призначений для керування шинними формувачами і діє впродовж усього циклу шини	Вихід (*z*)
$M/\overline{IO}$	*Memory/Input-Output* — вихідний сигнал ознаки звернення до пам'яті $(M/\overline{IO}=0)$ або зовнішнього пристрою $(M/\overline{IO}=0)$. Використовується для розподілу адресного простору пам'яті та введення-виведення	Вихід (*z*)
$\overline{WR}$	*Write* — вихід сигналу запису. Строб, який вказує на те, що МП виконує цикл запису в пам'ять або зовнішній пристрій і супроводжує дані, що видаються МП на шину даних	Вихід (*z*)
HOLD	*Hold* — вхід сигналу запиту захоплення шин від зовнішнього пристрою або контролера прямого доступу до пам'яті	Вхід
HLDA	*Hold Acknowledge* — вихідний сигнал підтвердження захоплення. Сигнал вказує на те, що МП перевів свої шини адреси-даних, адреси-стану і керування у *z*-стан	Вихід

Таблиця 2.5. **Призначення контактів МП *i*8086 у максимальиому режимі**

Позначення	Призначення	Тип
*ST*2, *ST*1, *ST*0 (*S*2 – *S*0)	Вихідні сигнали ліній стану; характеризують тип виконуваного циклу шини; використовуються для формування керуючих сигналів	Вихід (*z*)
$\overline{RQ}/\overline{GT0}$ $\overline{RQ}/\overline{GT1}$ ($\overline{RQ}/\overline{E0}$) ($\overline{RQ}/\overline{E1}$)	*Request-Grant* (*Request-Enable*) — два вхідних-вихідних сигнали запиту-надання локальної шини; використовуються для зв'язку з іншими процесорами, а саме, з арифметичним співпроцесором. Лінія *RQ*/*GT*1 має менший пріоритет	Вхід/вихід

Продовж. табл. 2.5

Позначення	Призначення	Тип
LOCK	*Lock* — вихідний сигнал блокування (зайнятості) шини — сигнал монополізації керування шиною; формується під час виконання команди за префіксом *LOCK* та інформує інші процесори і пристрої про те, що вони не повинні запрошувати системну шину	Вихід
*QS*1, *QS*0	*Queue Status* — два вихідних сигнали стану черги; ідентифікують стан внутрішньої шестибайтової черги команд МП і діють упродовж такту синхронізації після виконання операції над чергою. Сигнали *QS*1, *QS*0 призначені для співпроцесора, що контролює шину адреси-даних, фіксує момент, коли з програмної пам'яті вибирається призначена для нього команда з префіксом *ESC*, а після цього стежить за чергою команд і визначає момент, коли ця команда має виконуватися	Вихід

Таблиця 2.6. **Ідентифікація типу циклу шини**

Лінія стану			Тип циклу шини
*ST*2	*ST*1	*ST*0	
0	0	0	ПІДТВЕРДЖЕННЯ ПЕРЕРИВАННЯ ($\overline{INTA}$)
0	0	1	ВВЕДЕННЯ (хитання зовнішнього пристрою)
0	1	0	ВИВЕДЕННЯ (Запис у зовнішній пристрій)
0	1	1	ЗУПИНКА
1	0	0	ВИБІРКА команди
1	0	1	ЧИТАННЯ ПАМ'ЯТІ
1	1	0	ЗАПИС У ПАМ'ЯТЬ
1	1	1	Цикл шини відсутній

Таблиця 2.7. **Ідентифікація сегментного регістра**

*ST*4	*ST*3	Сегментний регістр	*ST*4	*ST*3	Сегментний регістр
0	0	*ES*	1	0	*CS*
0	1	*SS*	1	1	*DS*

Сигнал $ST2$ є логічним еквівалентом сигналу $M/\overline{IO}$, а $ST1$ — сигналу $DT/\overline{R}$ (див. розд. 2.1). Сигнали $ST4$, $ST3$ вказують, який сегментний регістр застосовується у цьому циклі (табл. 2.7) і можуть використовуватися для розширення адресного простору системи. Тоді окремий банк пам'яті ємністю 1 Мбайт виділяється кожному із чотирьох сегментів. До виводів МП $S4$, $S3$ підключають дешифратор, що вибирає відповідний банк пам'яті. Таке приймання забезпечує розширення адресної пам'яті до 4 Мбайт і захист від помилкового запису в сегмент, що перекривається з іншими сегментами.

Сигнал $ST5$ відповідає стану прапорця IF дозволу переривань: 0 — переривання заборонені, 1 — переривання дозволені. Сигнали $ST6$, $ST7$ не використовуються і зарезервовані для наступних МП.

Ідентифікація стану черги команд здіснюється за допомогою сигналів $QS1$, $QS2$ (див. табл. 2.5). Значення цих ліній визначає операцію над чергою команд (табл. 2.8).

Стан ліній запиту-надання локальної шини. Двонапрямлені лінії $\overline{RQ}/\overline{GT0}$, $\overline{RQ}/\overline{GT1}$ використовують для передавання імпульсних сигналів запиту-дозволу доступу до локальної шини. Процес доступу до шини здійснюється в такому порядку: спочатку пристрій, ввімкнений до локальної шини, який потребує доступу до загальних ресурсів, формує імпульс тривалістю один такт. Після цього наприкінці поточного циклу МП видає відповідний імпульс, що підтверджує можливість доступу до локальної шини. У наступному такті МП переводить шини адреси-даних і керування у високоімпедансний стан і відключається від каналу. Після закінчення роботи з каналом пристрій видає на ту саму лінію третій імпульс, що вказує на закінчення захоплення каналу. У наступному такті МП відновлює керування шиною і продовжує обчислення. Усі три імпульси мають однакову тривалість і низький активний рівень. Сигнали на лініях незалежні, однак лінія $\overline{RQ}/\overline{GT0}$

Таблиця 2.8. **Ідентифікація стану черги команд**

$QS1$	$QS0$	Операція над чергою команд
0	0	Операція відсутня, в останньому такті вибірки із черги не було
0	1	З черги вибрано перший байт команд
1	0	Черга порожня; була спустошена командою передавання керування
1	1	З черги вибрано наступний байт команди

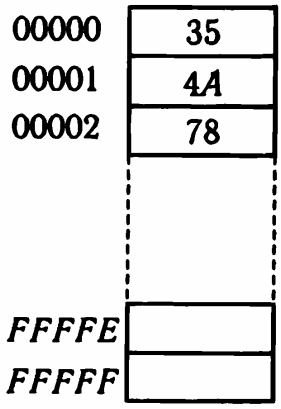


Рис. 2.13. Програмна модель пам'яті

має більш високий пріоритет, ніж лінія $\overline{RQ}/\overline{GT1}$, за одночасного надходження запитів. Кожна з двох розглянутих ліній використовується для встановлення режиму захоплення шин і є еквівалентною парі *HOLD* і *HLDA* МП *i*8086 у мінімальному режимі (див. розд. 2.1).

Організація пам'яті. Пам'ять — масив ємністю 1 Мбайт, тобто 2^{20} 8-розрядних комірок (рис. 2.13). У пам'яті зберігаються як байти, так і двобайтові слова. Слова розташовуються у двох сусідніх комірках пам'яті — старший байт зберігається у комірці зі старшою адресою, молодший — з молодшою. Адресою слова вважається адреса його молодшого байта. На рис. 2.13 подано приклад, коли за адресою 00000 зберігається байт 35*H*, а за адресою 00001 — слово 784A*H*. Початкові (00000*H* — 003*FFH*) і кінцеві адреси (*FFFF0H* — *FFFFFH*) зарезервовані відповідно для системи переривань та початкового встановлення.

Організація пам'яті, коли кожній адресі відповідає вміст однієї комірки пам'яті (див. рис. 2.13) називають *лінійною*. У МП *i*8086 застосовано *сегментну* організацію пам'яті, яка характеризується тим, що програмно-доступною є не вся пам'ять, а лише деякі сегменти, тобто області пам'яті. Усередині сегмента використовують лінійну адресацію.

Впровадження сегментної організації пам'яті можна пояснити так. Мікропроцесор *i*8086 — це 16-розрядний процесор, тобто він має 16-розрядну внутрішню шину, 16-розрядні регістри і суматори. Прагнення розробників ВІС адресувати якомога більший масив пам'яті зумовило використання 20-розрядної шини даних. Для порівняння: 16-розрядна шина адреси дає змогу адресувати 2^{16} = 64 Кбайт; а 20-розрядна -2^{20} = 1Мбайт.

Для формування 20-розрядної адреси у 16-розрядному процесорі використовують інформацію двох 16-розрядних регістрів. У МП *i*8086 20-розрядна адреса формується з двох 16-розрядних адрес, які називають *логічними*.

Перша логічна адреса, доповнена праворуч чотирма нулями, — це початкова адреса сегмента ємністю 64 Кбайт. Друга логічна адреса визначає зміщення у сегменті, визначаючи відстань від початку сегмента до адресованої комірки. Якщо вона має 0000, то адресується перша комірка сегмента, якщо *FFFFH*, — то остання. Отже, логічний адресний простір поділено на блоки суміжних адрес розміром 64 Кбайт, тобто сегменти.

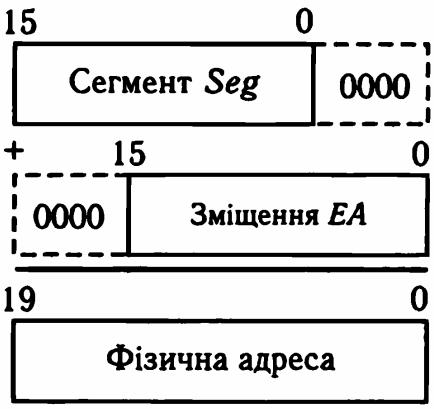


Рис. 2.14. Формування фізичної адреси

Такий підхід до організації пам'яті зручний ще й тому, що пам'ять зазвичай логічно поділяють на області коду (програмної пам'яті), даних і стеку. Фізична 20-розрядна адреса комірки пам'яті формується з двох 16-розрядних адрес — адреси сегмента *Seg* і виконавчої адреси *EA* (*Executive Address*), які додаються зі зміщенням на чотири розряди (рис. 2.14).

Зміщення адреси сегмента на чотири розряди ліворуч еквівалентне його множенню на 2^4. Тоді фізична адреса дорівнює $16 \times Seg + EA$. У вигляді першої логічної адреси *Seg* використовується вміст одного з чотирьох сегментних регістрів: *CS* (*Code Segment* — сегмент кодів), *DS* (*Data Segment* — сегмент даних), *ES* (*Extended Segment* — додатковий сегмент даних), *SS* (*Stack Segment* — сегмент стеку). Друга логічна адреса *EA*, або зміщення, залежить від сегмента. Так, у сегменті кодів як *EA* використовується вміст лічильника інструкцій *IP*, у сегментах даних значення *EA* залежить від засобу адресації операнда, в сегменті стеку використовуються регістри *SP* або *BP*.

Перетворення логічних адрес на фізичні є завжди однозначним, тобто парі *Seg* і *EA* відповідає єдина фізична адреса. Зворотне перетворення не є однозначним: фізичну адресу можна подати за допомогою 4096 пар логічних адрес.

У подальшому будемо позначати фізичну адресу у вигляді *Seg*:*EA*, де як *Seg* і *EA* можуть використовуватися як позначення регістрів, так і 16-розрядні дані.

Приклад 2.1. Знайти значення фізичної адреси за двома значеннями логічних адрес *CS*:*IP*.

Нехай вмістом сегментного регістра *CS* є число 2002 *H*, вмістом покажчика команд *IP* — 3175*H*. Додамо до значення *CS* чотири нулі праворуч:

$$CS(0000) = 0010\ 0000\ 0000\ 0010\ 0000B = 20020H.$$

Виконавши операцію додавання цієї величини до вмісту регістра *IP*, отримаємо фізичну адресу:

$$\begin{array}{r} 0010\ 0000\ 0000\ 0010\ 0000 \\ + \\ 0011\ 0001\ 0111\ 0101 \\ \hline 0010\ 0011\ 0001\ 1001\ 0101 \end{array} = 23195H.$$

Тому запис *CS:IP* за *CS* = 2002*H*, *IP* = 3175*H* відповідає фізичній адресі 23195 *H*.

Приклад 2.2. Знайти значення двох логічних адрес, які б відповідали фізичній адресі 23195*H* і не дорівнювали логічним адресам прикладу 2.1.

Значення фізичної адреси 23195*H* можна одержати додаванням двох інших логічних адрес 2100*H*: 2195*H*:

$$\begin{array}{r} 0010\ 0001\ 0000\ 0000\ 0000 \\ + \\ 0010\ 0001\ 1001\ 0101 \\ \hline 0010\ 0011\ 0001\ 1001\ 0101 \end{array} = 23195H.$$

Ємність пам'яті 1 Мбайт, починаючи з нульової адреси, розбивають на *параграфи* по 16 байт. Сегмент може починатися лише на межі параграфа, тобто в адресі сегмента молодші чотири біти адреси — нульові. Розміщення сегментів у пам'яті довільне: сегменти можуть частково або повністю перекриватися або не мати загальних частин. Змінюючи значення як першої, так і другої логічних адрес, можна адресувати будь-яку комірку із загальної пам'яті ємністю 1Мбайт.

На рис. 2.15, *а* показано розташування у просторі пам'яті 1 Мбайт чотирьох сегментів по 64 Кбайт без перекриття. Початкові адреси сегментів визначаються вмістом 16-розрядних сегментних регістрів, які доповнено праворуч чотирма нульовими бітами. Зміщення в сегменті кодів визначається вмістом регістра *IP*, зміщення в сегменті даних і додатковому сегменті даних — ефективною адресою *EA*, яка наводиться у команді, у сегменті стеку — вмістом регістра *SP*.

У сегментах кодів розташовано коди команд, тобто програма у машинних кодах, у решті сегментів — дані. Програма може звертатися лише до даних у сегментах (див. рис. 2.15), які позначені заштрихованими областями.

Змінюючи вміст сегментних регістрів, можна пересувати сегменти в межах усієї пам'яті 1 Мбайт. На рис. 2.15, *б* показано розташування сегментів кодів, даних, стеку та додаткового сегмента із частковим перекриттям. Це виникає тоді, коли вміст сегментних регістрів відрізняється менш ніж на 64 Кбайт/16 = 4096 байт

Програмна модель. Програмна модель МП *i*8086 (рис. 2.16) складається з РЗП, сегментних регістрів, покажчика команд і. регістра прапорців.

Регістри загального призначення поділяють на регістри даних і регістри-покажчики. До *регістрів даних* належать чотири 16-розрядних регістри: *AX*, *BX*, *CX*, *DX*. Кожний з цих регістрів складається з двох 8-розрядних регістрів, які можна незалежно адресувати за символічними іменами *AH*,

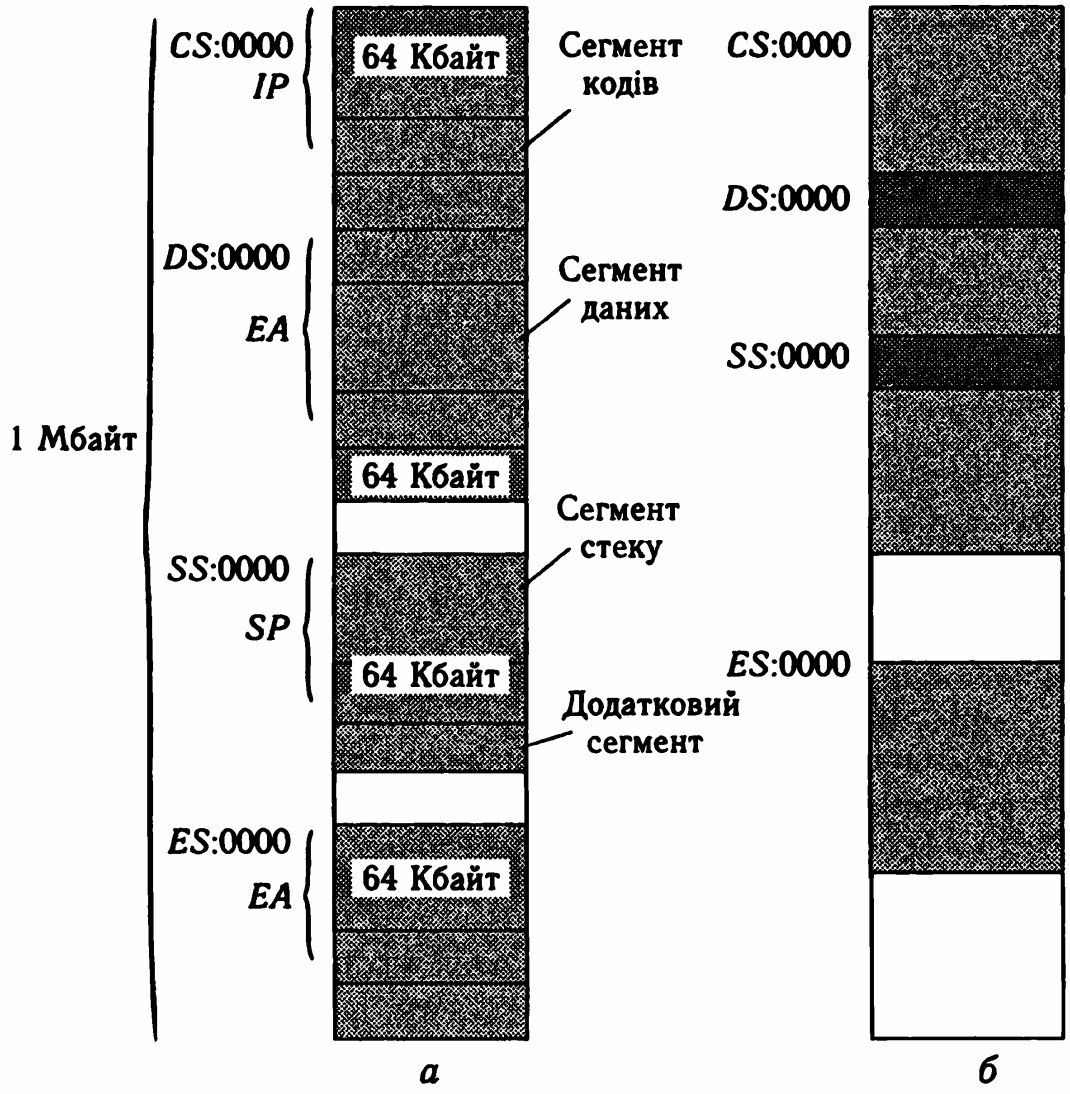


Рис. 2.15. Розташування сегментів у просторі пам'яті 1 Мбайт:
а — без перекриття; *б* — з частковим перекриттям

BH, *CH*, *DH* (старші байти — *High*) та *AL*, *BL*, *CL*, *DL* (молодші байти — *Low*). *Регістри-покажчики SP* (*Stack Pointer* — покажчик стеку), *BP* (*Base Pointer* — базовий регістр), *SI* (*Source Index* — індекс джерела), *DI* (*Destination Index* — індекс призначення) є 16-розрядними.Усі РЗП можна використати для зберігання даних, але в деяких командах припускається використання певного регістра за замовчуванням: *AX* — під час множення, ділення, введення та виведення слів; *AL* — під час множення, ділення, введення та виведення байтів, десяткової корекції, перетворення байтів (команда *XLAT*); *AH* — під час множення і ділення байтів; *BX* — під час трансляції; *CX* — як лічильник циклів і покажчик довжини рядків у рядкових командах; *CL* — для зберігання зміщення з указанням змінної; *DX* — під час множення і ділення слів, введення і виведення з непрямою адресацією; *SP* — під час

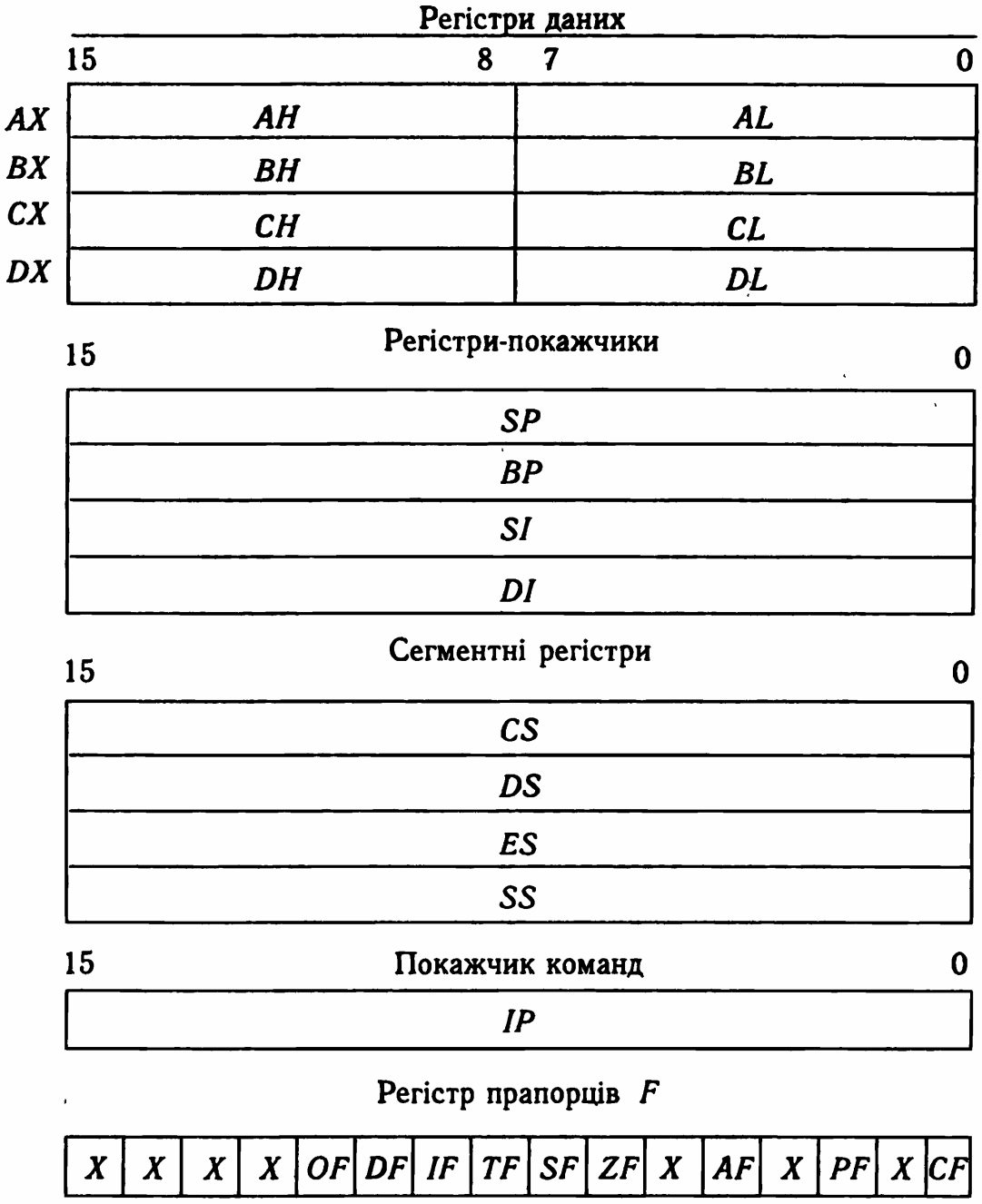


Рис. 2.16. Програмна модель МП *i*8086

операцій зі стеком; *SI*, *DI* — під час рядкових операцій. На відміну від 8-розрядних МП регістр *SP* зберігає зміщення останньої зайнятої комірки стеку щодо початку сегмента стеку, а повна адреса стеку визначається як *SS* : *SP*.

Сегментні регістри CS, *DS*, *ES*, *SS* визначають початкові адреси чотирьох сегментів пам'яті. Використання сегментних регістрів визначається типом звернення до пам'яті (табл. 2.9).

Для деяких типів звернень допускається заміна сегментного регістра за замовчуванням на альтернативний, яка реалізується префіксами команд *CS:*, *DS:*, *SS:*, *ES:*.

Таблиця 2.9. **Використаиня регістрів під час адресації пам'яті**

Тип звернення до пам'яті	Сегментний регістр		Зміщення
	за замовчуванням	альтернативний	
Вибірка команд	*CS*	Немає	*IP*
Стекові операції	*SS*	Немає	*SP*
Адресація змінної	*DS*	*CS, ES, SS*	*EA*
Рядок-джерело*	*DS*	*CS, ES, SS*	*SI*
Рядок-приймач*	*ES*	Немає	*DI*
Використання *BP* під час звернення до стеку у процесі читання/запису даних	*SS*	*CS, ES, DS*	*EA*

*Примітка. Рядок-джерело і рядок-приймач — це рядки даних (масиви), які беруть участь у рядкових командах.

Приклад 2.3. Переслати вміст комірки пам'яті з адресою *DS*:1000*H* в регістр-акумулятор *AL*.

Для того, щоб переслати вміст комірки пам'яті в акумулятор, треба використати команду пересилки *MOV dst, src*, де операндом призначення *dst* (*Destination*) є регістр *AL*, а операндом джерела інформації *src* (*Source*) — комірка пам'яті. Комірка пам'яті позначається квадратними дужками, всередині яких записується зміщення у сегменті, тобто друга логічна адреса. Перша логічна адреса за замовчуванням є вмістом регістра *DS*.

Після запису мнемоніки команди пересилання *MOV* записується операнд-призначення, а потім через кому — операнд-джерело. Після операндів через крапку з комою записується коментар до команди. Отже, за командою

$$MOV\ AL,\ [1000H];\ AL \leftarrow DS:[1000H]$$

у регістр *AL* пересилається байт з комірки пам'яті з адресою *DS*:1000*H*.

Зазначимо, що перед використанням цієї команди вміст регістра *DS* має бути визначеним.

Приклад 2.4. Переслати вміст комірки пам'яті з адресою *ES*:1000*H* в регістр-акумулятор *AL*.

За командою з префіксом *ES*

$$MOV\ AL,\ ES:\ [1000H];\ AL \leftarrow ES:[1000H]$$

в *AL* пересилається вміст комірки пам'яті з адресою *ES*: 1000*H*.

На відміну від 8-розрядних МП покажчик команд *IP* зберігає зміщення в сегменті кодів поточної команди.

Регістр прапорців зберігає ознаки результатів виконання арифметичних і логічних операцій та керуючі ознаки, які можна встановити або скинути програмно. Типи прапорців подано в табл. 2.10.

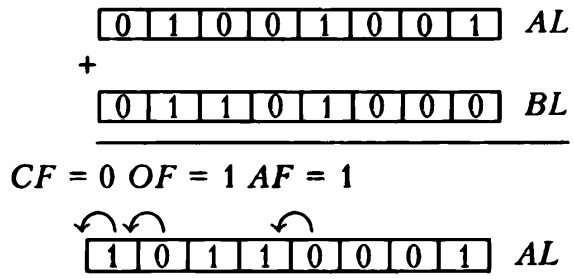
Приклад 2.5. Визначити значення прапорців після виконання команди *ADD AL, BL* (додавання вмісту 8-розрядних регістрів *AL, BL;*

Таблиця 2.10. **Призначення різних типів прапорців**

Позначення прапорця	Призначення прапорця	Розрядність операнда	
		8	16
AF	*Auxiliary Flag* — прапорець допоміжного перенесення-позики з молодшої тетради у старшу (з розряду *D*3 у розряд *D*4). Використовується за десяткової арифметики	+	–
CF	*Carry Flag* — прапорець перенесення-позики. Встановлюється у разі виходу результату додавання (віднімання) беззнакових операндів за межу діапазону. У командах зсуву прапорець *CF* фіксує значення старшого біта	+	+
OF	*Overflow Flag* — прапорець переповнення, встановлюється у разі виходу знакового результату за межу діапазону	+	+
SF	*Sign Flag* — прапорець знака. Дублює значення старшого біта результату; *SF* = = 0 для позитивних чисел і *SF* = 1 — для негативних	+	+
PF	*Parity Flag* — прапорець паритету (парності). Встановлюється за парного числа одиниць у результаті	+	–
ZF	*Zero Flag* — прапорець нульового результату. Встановлюється у разі отримання нульового результату операції	+	+
DF	*Direction Flag* — прапорець керування напрямом у рядкових операціях. За *DF* = = 1 індексні регістри *SI*, *DI*, що беруть участь у рядкових операціях, автоматично декрементуються на кількість байтів операнда, за *DF* = 0 — інкрементуються		
IF	*Interrupt-enable Flag* — прапорець дозволу переривань. За *IF* = 1 дозволяється виконання маскованих апаратних переривань		
TF	*Trap Flag* — прапорець трасування (покрокового режиму). Після його встановлення і виконання кожної команди відбувається внутрішнє переривання 1 (*INT* 1)		

результат передається в AL), якщо в регістрі AL міститься число $49H$, а в регістрі BL — $68H$.

Після виконання команди додавання прапорці встановлюють так:



$AF = 1$, $CF = 0$, $OF = 1$, $SF = 1$, $PF = 1$, $ZF = 0$.

Пояснимо встановлення прапорців. Результат операції додавання не є нульовим, тому прапорець ZF скинуто, тобто $ZF = 0$. Старший розряд результату дорівнює одиниці, тому $SF = 1$. Кількість одиниць у результаті 4, тобто парне число, отже, $PF = 1$. Після додавання виникло переповнення з молодшої тетради у старшу ($AF = 1$), у знаковий розряд ($OF = 1$). Переповнення розрядної сітки не відбулося, тому $CF = 0$.

Адресація портів введення-виведення. Простір адрес портів введення-виведення несегментований, займає 64 Кбайт і адресується 16 молодшими розрядами 20-розрядної шини адреси. Порти можуть бути як 8-, так і 16-розрядними. Будь-які два суміжних 8-розрядних порти можна вважати 16-розрядним портом аналогічно слову в пам'яті. При цьому для обміну з 8-розрядними портами використовується регістр AL, а з 16-розрядними — регістр AX. Перші 256 портів (з номерами $0-0FFH$) можна адресувати за допомогою прямої адресації.

Приклад 2.6. Ввести інформацію з 8-розрядного порту з адресою $56H$ у регістр-акумулятор AL.

Для того щоб ввести інформацію з 8-розрядного порту з адресою $56H$ в акумулятор AL, треба виконати команду введення IN (введення). Першим операндом команди є позначення акумулятора AL, якщо вводиться байт інформації, або AX, якщо вводиться слово. У цьому випадку треба використати операнд AL. Другим операндом є номер порту $56H$. Тому за командою

$$IN\ AL,\ 56H \qquad ;\ AL \leftarrow P_8(56H)$$

відбудеться введення інформації з 8-розрядного порту з адресою $56H$ до акумулятора.

Зазначимо, що допускається позначення номера порту у квадратних дужках:

$$IN\ AL,\ [56H] \qquad ;\ AL \leftarrow P_8(56H).$$

Приклад 2.7. Вивести інформацію з регістра-акумулятора AX у 16-розрядний порт з адресою $34H$.

Для того щоб вивести інформацію з акумулятора *AX* до 16-розрядного порту з адресою 34*H*, треба виконати команду виведення *OUT*. Першим операндом команди є номер порту 34*H*, другим — позначення акумулятора *AL*, якщо виводиться байт інформації, або *AX*, якщо виводиться слово. У цьому разі треба використати операнд *AX*. Отже, за командою

$$OUT\ 34H,\ AX \qquad ;\ AX \rightarrow P_{16}(34H)$$

відбудеться виведення інформації з регістра *AX* на 16-розрядний порт з адресою 34*H*.

Усі 64 Кбайт портів адресуються непрямо — за допомогою регістра *DX*.

Приклад 2.8. Ввести інформацію з 8-розрядного порту з адресою, що знаходиться у регістрі *DX*, у регістр-аккумулятор *AL*.

Для того щоб ввести інформацію, треба виконати команду введення *IN*, першим операндом якої є позначення акумулятора *AL*, а другим — позначення регістра *DX*.

Тому за командою

$$IN\ AL,\ DX \qquad ;\ AL \leftarrow P_8(DX)$$

відбудеться введення інформації в акумулятор *AL* з 8-розрядного порту з адресою, що знаходиться в регістрі *DX*. Вміст регістра *DX* має бути визначений до моменту виконання команди введення.

Допускається запис команди з непрямою адресацією у вигляді

$$IN\ AL,\ [DX]\ .$$

Типи адресації операндів. У МП *i*8086 використовуються ті самі основні типи адресації — пряма, регістрова, безпосередня і непряма, що й для 8-розрядних процесорів (див. п. 2.1), однак непряма адресація має такі різновиди: базова, індексна, базово-індексна.

Базова адресація. Ефективна адреса операнда *EA* обчислюється складанням вмісту базових регістрів *BX* або *BP* і зміщення (8- або 16-розрядного знакового числа). У деяких випадках зміщення може не відбуватися.

Приклад 2.9. Переслати в регістр-акумулятор *AX* вміст комірки пам'яті, що розташована у сегменті даних і має ефективну адресу (зміщення у сегменті), яка дорівнює сумі вмісту регістра *BX* і числа 2000*H*.

Для того щоб переслати вміст комірки пам'яті в акумулятор, треба використати команду пересилання *MOV dst, src*, де операндом призначення *dst* (*destination*) є регістр *AX*, а операндом джерела інформації *src* (*source*) — комірка пам'яті. Комірка пам'яті позначається квадратними дужками, а всередині записується значення ефективної адреси, тобто *BX* + 2000*H*.

Отже, за командою

$$MOV\ AX,\ [BX + 2000H] \qquad ;\ AX \leftarrow DS{:}[BX + 2000H]$$

у регістр *AX* пересилається байт з комірки пам'яті з адресою *DS*: *BX* + + 2000*H*.

Зазначимо, що перед використанням цієї команди вмісти регістрів *DS* і *BX* визначаються заздалегідь.

Індексна адресація. За індексної адресації як адреси зміщення використовують вміст індексних регістрів *SI* або *DI* та зміщення.

Приклад 2.10. Переслати у регістр-акумулятор *AX* вміст комірки пам'яті, розташованого у сегменті даних з ефективною адресою (зміщення у сегменті), яка дорівнює сумі вмісту регістра *SI* і числа 5000*H*.

За командою

MOV AX, [*SI* + 5000*H*] ; *AX* ← *DS*: [*SI* + 5000*H*]

у регістр *AX* пересилається байт з комірки пам'яті з адресою *DS*: *SI* + + 5000*H*.

Перед використанням цієї команди вмісти регістрів *DS* і *SI* визначають заздалегідь.

Базово-індексна адресація. Ефективна адреса операнда *EA* дорівнює сумі вмісту базових регістрів *BX* або *BP*, індексних регістрів *SI* або *DI* та зміщення.

Приклад 2.11. Переслати у регістр-акумулятор *AX* вміст комірки пам'яті, що розташована у сегменті даних і має ефективну адресу, яка дорівнює сумі вмісту двох регістрів *SI* і *BX*.

За командою

MOV AX, [*SI* + *BX*] ; *AX* ← *DS*: [*SI* + *BX*]

у регістр *AX* перссилається байт з комірки пам'яті з адресою *DS*: *SI* + + *BX*.

Перед використанням цієї команди вмісти регістрів *DS*, *SI* і *BX* визначають заздалегідь.

Базову та індексну адресацію застосовують для звернення до елементів одновимірного, а базово-індексну — до двовимірного масиву.

Цикли шини процесора. Впродовж *циклу шини* МП виставляє адресу комірки пам'яті або ПВВ на шину адреси, формує керуючі сигнали читання-запису, а після цього зчитує або записує дані. Крім циклів ЧИТАННЯ і ЗАПИСУ ПАМ'ЯТІ або ПВВ існують цикли ПІДТВЕРДЖЕННЯ-ПЕРЕРИВАННЯ і ЗАХОПЛЕННЯ ШИН. Цикл шини може ініціювати не лише незалежний процесор *i*8086, а й арифметичний співпроцесор або співпроцесор введення-виведення. Розрізняють цикли шини в мінімальному і максимальному режимах. Часові діаграми циклів ЧИТАННЯ та ЗАПИСУ у мінімальному режимі зображено на рис. 2.17.

Цикл шини складається як мінімум із чотирьох тактів. Такт визначається як проміжок часу між задніми фронтами двох сусідніх імпульсів *CLK*. Будь-який цикл шини може бути необмежено розтягнуто за допомогою сигналу готовності *READY*; при цьому процесор вводить необов'язкові такти очікування T_w.

Цикли звернення до порту відрізняються від циклів пам'яті тим, що старші розряди шини адреси мають нульове значен-

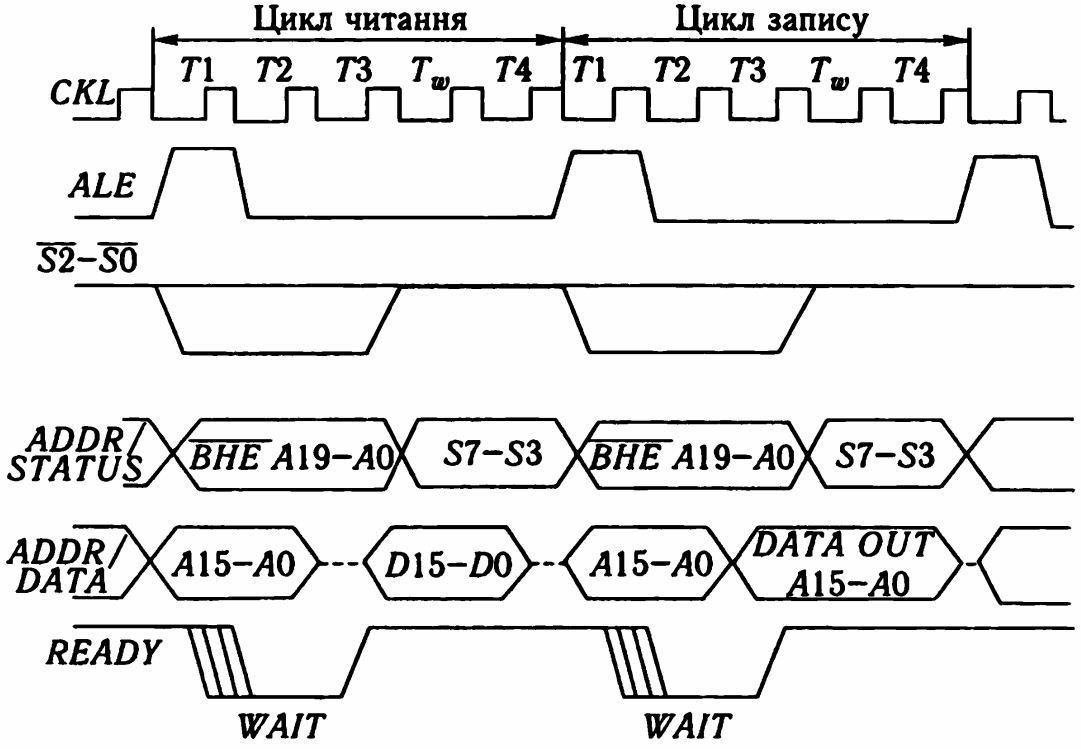


Рис. 2.17. Цикли ЧИТАННЯ і ЗАПИСУ МП *i*8086 у мінімальному режимі

ня (за непрямої адресації за допомогою *DX* сигнали на лініях *A*19—*A*16 набувають значень *L*-рівня, за прямої — сигнали на лініях *A*19—*A*8 є нульовими).

Цикл ПІДТВЕРДЖЕННЯ-ПЕРЕРИВАННЯ формується аналогічно циклу ЧИТАННЯ порту, але замість активного сигналу $\overline{IOR}$ активним є сигнал $\overline{INTA}$, а шина адреси процесором не керується.

Типи переривань. Процесор *i*8086 може обробляти до 256 типів переривань. Кожному перериванню відповідає свій вектор — подвійне слово, що містить адресу *CS*: *IP* підпрограми, що викликається. Під вектори переривань у загальному просторі адрес пам'яті відводиться 1 Кбайт, починаючи з нульової адреси (рис. 2.18).

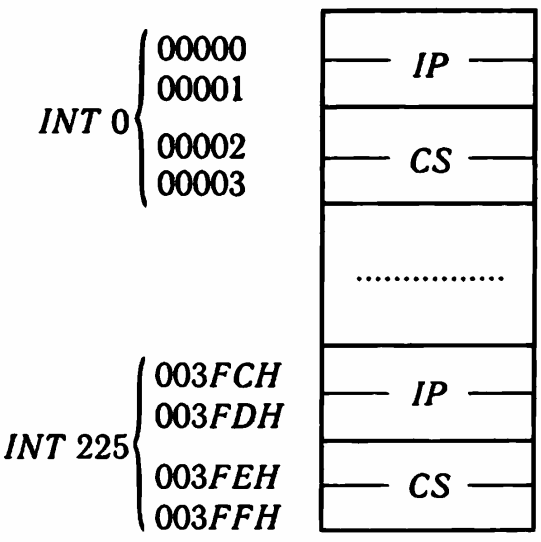


Рис. 2.18. Карта векторів переривань

Під час переходу на підпрограму обробки переривань *INT n* (*n* — тип перери-

вання) процесор переміщує у стек вміст регістрів *IP*, *CS*, регістр прапорців *F* і скидає прапорець дозволу переривання *IF*; обчислює адресу $4 \times n$ і перше слово за цією адресою переміщує у *IP*, друге — у *CS*. Послідовність цих дій еквівалентна командам:

```
PUSHF                ; Запам'ятовування у стеку пра-
                     ; порців
CALL FAR i_proc_4n   ; Далекий виклик підпрограми об-
                     ; робки
                     ; переривання
```

Скидання прапорця переривання *IF* не дозволяє перервати виконання підпрограми обробки переривання до її завершення або виконання команди дозволу *STI*. Останньою командою підпрограми обробки переривання є команда *IRET*. За цією командою процесор вибирає зі стеку адресу повернення (адресу команди, наступної за командою *INT*) і вміст регістра прапорців.

Типи переривань показано на рис. 2.19. Переривання поділяють на *зовнішні апаратні* та *внутрішні*. Запити *IRQ* і зовнішніх апаратних переривань надходять до системи переривань або на вивід немаскованого переривання *NMI* МП. Система переривання формує сигнал *INTR* маскованого переривання МП. Зазначимо, що масковане переривання відрізняється від немаскованого тим, що перше може бути заборонено програмно — командою скидання прапорця дозво-

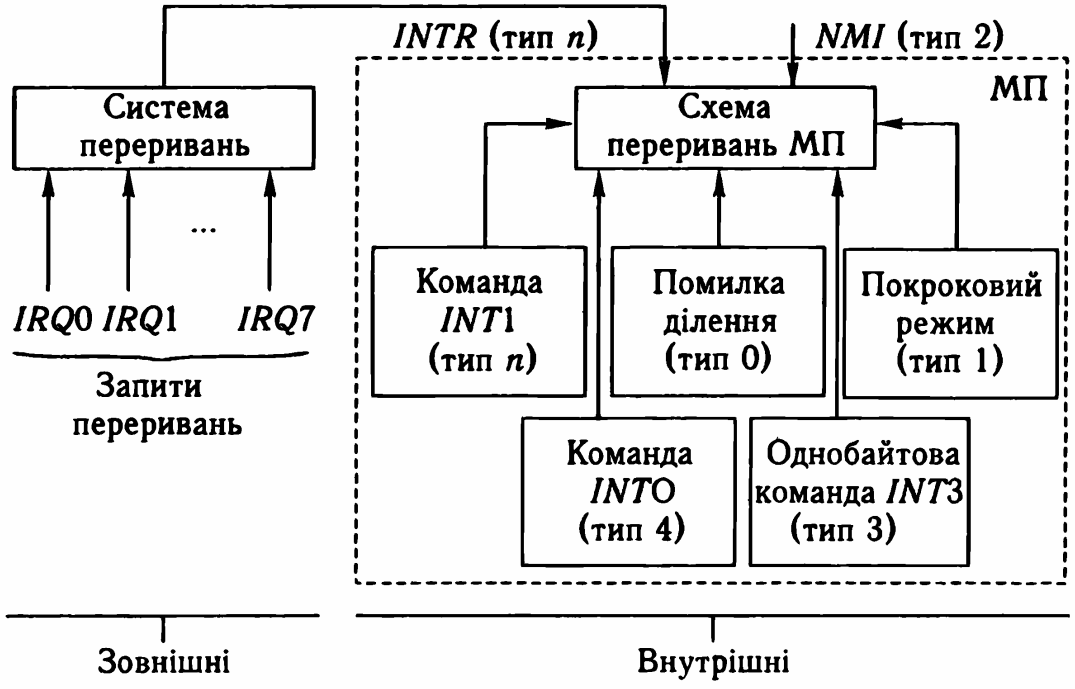


Рис. 2.19. Типи переривань

лу переривань *IF*. У цьому разі під час надходження запитів переривання вони будуть ігноруватися. Внутрішні переривання процесора поділяють на *програмні* та *апаратні*. Джерелами *внутрішніх програмних* переривань (рис. 2.19) є помилка ділення (тип 0), покроковий режим (тип 1) і команда *INTO* (тип 4).

Внутрішні програмні переривання *INT n* та *INT* 3 виконуються за командами переривання і дозволяють викликати підпрограми обробки переривань (наприклад, сервісні підпрограми *BIOS* і *DOS*) без застосування дальніх викликів. На відміну від *INT n*, переривання *INT* 3 є однобайтовою командою і зазвичай використовується для передавання керування підпрограмі-налагоджувачу. Слід зазначити, що виконання програмних переривань не залежить від прапорця дозволу переривань *IF*.

Внутрішні апаратні переривання процесора виникають у таких випадках:

- під час ділення на нуль (тип 0);
- за встановленого прапорця трасування (тип 1). У цьому разі переривання відбувається після виконання кожної команди;
- після команди *INTO* (тип 4), якщо встановлений прапорець переповнення *OF*.

Апаратні переривання виникають у разі активного рівня сигналів на контактах МП-*NMI* (немасковане переривання — тип 2) і *INTR* (масковані, переривання типи 5—255). Маскові переривання виконуються за встановленого прапорця *IF*. Під час переходу до підпрограми обробки апаратного переривання процесор формує два цикли підтвердження переривання один за одним, в яких генерується сигнал $\overline{INTA}$. За другим імпульсом $\overline{INTA}$ контролер переривань передає по шині даних номер вектора переривання *n*. Далі дії процесора аналогічні виконанню програмного переривання. Обробка поточного переривання може бути перервана немаскованим перериванням або іншим маскованим перериванням вищого пріоритету тоді, якщо підпрограма-обробник встановить прапорець дозволу переривання *IF*. Немасковане переривання виконується незалежно від стану прапорця *IF*.

Мікропроцесор *i*8088 відрізняється від мікропроцесора *i*8086 тим, що має зовнішню 8-розрядну шину даних за наявності внутрішньої 16-розрядної шини. Зменшення розрядності шини даних спрощує побудову блоків пам'яті інтерфейсу із зовнішніми пристроями, але продуктивність процесора знижується на 20—30 %. Структурна схема МП *i*8088 аналогічна схемі МП *i*8086, однак довжина черги команд скорочена до 4 байт, а попередня вибірка виконується за наяв-

ності одного вільного байта. Ці властивості оптимізують конвеєр з урахуванням розрядності шини. З погляду програмного забезпечення процесори ідентичні, їхня система команд і набір регістрів однакові. Так само, як і МП *i*8086, МП *i*8088 виконує 8- і 16-розрядні логічні та арифметичні операції, включаючи множення й ділення в двійковому та двійково-десятковому кодах, операції з рядками, підтримує режими переривання, прямого доступу до пам'яті, операції з портами. Розташування контактів МП *i*8088 і *i*8086 збігається, за винятком того, що лінії $AD15-AD8$ використовуються лише для адреси, а лінія $\overline{BHE}$ замінена лінією стану $ST0$. Сигнали $ST0$, $DT/\bar{R}$ і $IO/\bar{M}$ можуть бути використані для ідентифікації циклу шини (див. табл. 2.6).

Мікропроцесори *i*80186(*i*80188) є програмно сумісними з МП *i*8086. Розрядність шини адреси — 20, шини даних — відповідно 16 і 8. Процесори мають вбудовані периферійні контролери переривань, прямого доступу до пам'яті, триканальний таймер і тактовий генератор. Мікропроцесори *i*80C186/*i*80C188 обладнані засобами керування енергоспоживанням. Є модифікації МП з вбудованим послідовним портом та контролерами динамічної пам'яті.

2.3. Система команд мікропроцесора *i*8086

Система команд мікропроцесора *i*8086 (табл. 2.11) містить 91 мнемокод. Усі команди МП поділяють на п'ять груп:

- команди передавання інформації (команди пересилання, роботи зі стеком, введення-виведення);
- команди оброблення інформації (арифметичні, логічні, команди зсуву);
- рядкові команди;
- команди передавання керування, включаючи команди переривань;
- команди керування станом МП.

У табл. 2.11 використано такі позначення:

src — операнд-джерело;
dest — операнд-призначення;
reg — 8-/16-розрядний РЗП;
*reg*8 — 8-розрядний РЗП;
*reg*16 — 16-розрядний РЗП;
sr — сегментний регістр;
mem — 8-/16-розрядна комірка пам'яті;
*mem*8 — 8-розрядна комірка пам'яті;

*mem*16 — 16-розрядна комірка пам'яті;

r/m — 8-/16-розрядний регістр або комірка пам'яті;

r/m/i — 8-/16-розрядний регістр, комірка пам'яті або безпосередній операнд;

immed — безпосередній операнд;

disp — 8-/16-розрядне зміщення під час задання адреси;

*disp*8 — 8-розрядне зміщення;

*disp*16 — 16-розрядне зміщення;

target — мітка, до якої здійснюється перехід;

seg target — перша логічна адреса (сегментний адрес) мітки *target*;

offset target — друга логічна адреса (зміщення у сегменті) мітки *target*;

A — акумулятор *AL* або *AX*;

m[*disp*] — комірка пам'яті з ефективною адресою *EA* = *disp*.

Значення кількості тактів *EA*, необхідне для обчислення ефективної адреси, залежить від способу адресації операнда (табл. 2.12).

Таблиця 2.11. **Система команд мікропроцесора *i*8086**

Мнемокод команди	Опис команди	Алгоритм команди	Кількість байт	Кількість тактів
КОМАНДИ ПЕРЕДАВАННЯ ІНФОРМАЦІЇ				
Команди пересилання				
MOV dest, src	Пересилання даних з регістра, комірки пам'яті або безпосереднього операнда у регістр або пам'ять	*reg* ← *reg*	2	2
		sr ← *reg*	2	2
		reg ← *sr*	2	2
		mem ← *reg*	2–4*1	9 + *EA*
		reg ← *mem*	2–4*1	8 + *EA*
		mem ← *sr*	2–4*1	9 + *EA*
		sr ← *mem*	2–4*1	8 + *EA*
		a ← *mem*	3	11
		mem ← *a*	3	11
		*mem*8 ← *immed*	3–5*2	10 + *EA*
		*mem*16 ← *immed*	4–6*3	10 + *EA*
		*reg*8 ← *immed*	2	4
		*reg*16 ← *immed*	3	4
XCHG r/m, reg	Обмін даними між регістрами або регістром і пам'яттю	*reg* ←→ *reg*	2	4
		mem ←→ *reg*	2–4*1	17 + *EA*
		A ←→ *reg*	1	3
XLAT	Перекодування вмісту *AL* у значення байта пам'яті з адресою *ES*: [*BX* + (*AL*)]	*AL* ← *ES*: [*BX* + + (*AL*)]	1	11

Мнемокод команди	Опис команди	Алгоритм команди	Кількість байт	Кількість тактів
*LEA reg*16, *mem*	Завантаження ефективної адреси комірки пам'яті *mem* у регістр	*reg* ← *EA*	2–4*1	2 + *EA*
*LDS reg*16, *mem*	Завантаження у регістр *reg*16 слова із комірки пам'яті за адресою [*mem*], у *DS* – наступного слова з комірки за адресою [*mem* + 2]	*reg* ← [*mem*] *DS* ← [*mem*+2]	2–4*1	16 + *EA*
*LES reg*16, *mem*	Завантаження у регістр *reg*16 слова із комірки пам'яті за адресою [*mem*], в *ES* – наступного слова з комірки за адресою [*mem* + 2]	*reg* ← [*mem*] *ES* ← [*mem* + 2]	2–4*1	16 + *EA*
LAHF	Завантаження молодшого байта регістра прапорців *FL* в *AH*	*AH* ← *FL*	1	4
SAHF	Збереження *AH* у молодшому байті регістра прапорців *FL*	*FL* ← *AH*	1	4
Команди роботи зі стеком				
PUSH r/m/sr	Пересилання слова з регістра або з пам'яті у стек	*SP* ← *SP* – 2 [*SS*:*SP*] ← *r/m*	2–4*1	11/(16 + + *EA*)
		SP ← *SP* – 2 [*SS*:*SP*] ← *sr*	1	10
PUSHF	Пересилання у стек вмісту регістра прапорців	*SP* ← *SP* – 2 [*SS*:*SP*] ← *F*	1	10
POP r/m/sr	Пересилання слова даних зі стеку в регістр або пам'ять	*r/m* ← [*SS*:*SP*] *SP* ← *SP* + 2	2–4*1	8/(16 + + *EA*)
		sr ← [*SS*:*SP*] *SP* ← *SP*+2	1	8
POPF	Пересилання даних зі стеку в регістр прапорців	*F* ← [*SS*:*SP*] *SP* ← *SP* + 2	1	8

Продовж. табл. 2.11

Мнемокод команди	Опис команди	Алгоритм команди	Кількість байт	Кількість тактів
		Команди введення-виведення		
IN AL, P8 *IN AL, DX*	Введення в акумулятор *AL* байта з 8-розрядного порту з адресою *P8* або з адресою, що зберігається в *DX*	*AL* ← Порт (*P8*) *AL* ← Порт (*DX*)	2 1	10 8
IN AX, P8 *IN AX, DX*	Введення в акумулятор *AX* слова із 16-розрядного порту з адресою *P8* або з адресою, що зберігається в *DX*	*AX* ← Порт (*P8*) *AX* ← Порт (*DX*)	2 1	10 8
OUT P8, AL *OUT DX, AL*	Виведення байта з акумулятора *AL* у 8-розрядний порт з адресою *P8* або з адресою, що зберігається в *DX*	Порт (*P8*) ← *AL* Порт (*DX*) ← *AL*	2 1	10 8
OUT P8, AX *OUT DX, AX*	Виведення слова з акумулятора *AX* у 16-розрядний порт з адресою *P8* або з адресою, що зберігається в *DX*	Порт (*P8*) ← *AX* Порт (*DX*) ← *AX*	2 1	10 8
		КОМАНДИ ОБРОБЛЕННЯ ІНФОРМАЦІЇ		
		Арифметичні команди		
ADD r/m, r/m/i	Додавання двох операндів	*r/reg* ← *r/reg* + *reg* *reg* ← *reg* + *r/m* *reg*8 ← *reg*8 + *immed* *reg*16 ← *reg*16 + + *immed* *mem*8 ← *mem*8 + + *immed* *mem*16 ← *mem*16 + + *immed* *AL* ← *AL* + *immed* *AX* ← *AX* + *immed*	2 2–4*1 2–4*1 3 4 3–5*2 2 3	3 16 + *EA* 9 + *EA* 4 4 17 + *EA* 4 4
ADC r/m, r/m/i	Додавання двох операндів і прапорця перенесення *CF* від попередньої операції. Прапорець перене-	*reg* ← *reg* + *reg* + *CF* *mem* ← *mem* + + *reg* + *CF* *reg* ← *reg* + + *mem* + *CF*	2 2–4*1 2–4*1	3 16 + *EA* 9 + *EA*

Продовж. табл. 2.11

Мнемокод команди	Опис команди	Алгоритм команди	Кіль-кість байт	Кіль-кість тактів
	сення додається до молодшого біта результату	*reg*8 ← *reg*8 + + *immed* + *CF*	3	4
		*reg*16 ← *reg*16 + + *immed* + *CF*	4	4
		*mem*8 ← *mem*8 + + *immed* + *CF*	3–5*2	17 + *EA*
		mem16 ← *mem16* + + *imm* + *CF*	4–6*3	17 + *EA*
		AL ← *AL* + + *immed* + *CF*	2	4
		AX ← *AX* + + *immed* + *CF*	3	4
INC r/m	Інкремент (додавання з одиницею). Команда не діє на прапорець *CF*	*reg*8 ← *reg*8 + 1	2	3
		*reg*16 ← *reg*16 + 1	1	2
		mem ← *mem*+1	2–4*1	15 + *EA*
AAA	Корекція після додавання розпакованих двійково-десяткових чисел		1	4
DAA	Корекція після додавання упакованих двійково-десяткових чиссл		1	4
SUB r/m, r/m/i	Віднімання двох операндів	*reg* ← *reg* − *reg*	2	3
		mem ← *mem* − *reg*	2–4*1	16 + *EA*
		reg ← *reg* − *mem*	2–4*1	9 + *EA*
		reg8 ← *reg* − *immed*	3	4
		reg16 ← *reg1* − − *immed*	4	4
		mem8 ← *mem* − − *immed*	3–5*2	17 + *EA*
		mem16 ← *mem1* − − *immed*	4–6*3	17 + *EA*
		AL ← *A* − *immed*	2	4
		AX ← *A* − *immed*	3	4
SBB r/m, r/m/i	Віднімання байта та прапорця позики *CF* від попередньої операції. Прапорець позики віднімається від молодшого біта результату	*reg* ← *reg* − *reg* − *CF*	2	3
		mem ← *mem* − − *reg* − *CF*	2–4*1	16 + *EA*
		reg ← *reg* − *mem* − *CF*	2–4*1	9 + *EA*
		reg8 ← *reg8* − − *immed* − *CF*	3	4
		*reg*16 ← *reg16* − − *immed* − *CF*	4	4
		mem8 ← *mem8* − − *immed* − *CF*	3–5*2	17 + *EA*
		mem16 ← *mem16* − − *immed* −*CF*	4–6*3	17 + *EA*

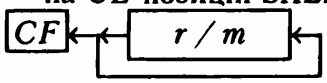
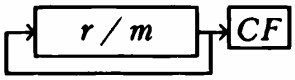
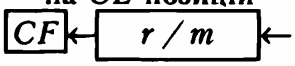
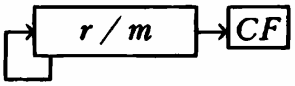
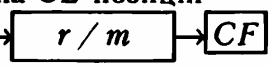
Продовж. табл. 2.11

Мнемокод команди	Опис команди	Алгоритм команди	Кількість байт	Кількість тактів
		$AL \leftarrow AL -$ $- immed - CF$	2	4
		$AX \leftarrow AX -$ $- immed - CF$	3	4
DEC r/m	Декремент (віднімання одиниці). Команда не діє на прапорець *CF*	$reg8 \leftarrow reg8 - 1$	2	3
		$reg16 \leftarrow reg16 - 1$	1	2
		$mem \leftarrow mem - 1$	2–4*1	15 + *EA*
NEG r/m	Зміна знака операнда	$reg \leftarrow - reg$	3	3
		$mem \leftarrow - mem$	2–4*1	16 + *EA*
CMP r/m, r/m/i	Порівняння двох операндів – встановлення вмісту регістра прапорів *F* за результатом віднімання (без збереження результату віднімання)	$F \Leftarrow reg - reg$	2	3
		$F \Leftarrow mem - reg$	2–4*1	9 + *EA*
		$F \Leftarrow reg - mem$	2–4*1	9 + *EA*
		$F \Leftarrow reg8 - immed$	3	4
		$F \Leftarrow reg16 - immed$	4	4
		$F \Leftarrow mem8 - immed$	3–5*2	1 + *EA*
		$F \Leftarrow mem16 - imm$	4–6*3	1 + *EA*
		$F \Leftarrow AL - immed$	2	4
		$F \Leftarrow AX - immed$	3	4
AAS	Корекція після віднімання розпакованих двійково-десяткових чисел		1	4
DAS	Корекція після віднімання упакованих двійково-десяткових чисел		1	4
MUL r/m	Множення *AL* (*AX*) на беззнакове значення *r/m*	$AX \leftarrow \|AL\| \times \|reg8\|$	2	70–77
		$(DX, AX) \leftarrow \|AX\| \times$ $\times \|reg16\|$	2	118–133
		$AX \leftarrow \|AL\| \times \|mem8\|$	2–4*1	76–83 + + *EA*
		$(DX, AX) \leftarrow \|AX\| \times$ $\times \|mem16\|$	2–4*1	124–139+ + *EA*
IMUL r/m	Множення *AL* (*AX*) на знакове значення *r/m*	$AX \leftarrow \|AL\| \times \|reg8\|$	2	80 – 98
		$(DX, AX) \leftarrow \|AX\| \times$ $\times \|reg16\|$	2	128 – 154
		$AX \leftarrow \|AL\| \times \|mem8\|$	2–4*1	96–104+ + *EA*
		$(DX, AX) \leftarrow \|AX\| \times$ $\times \|mem16\|$	2–4*1	134–160+ + *EA*
AAM	Корекція після множення розпакованих двійково-десяткових чиссл		2	83

Мнемокод команди	Опис команди	Алгоритм команди	Кіль-кість байт	Кіль-кість тактів
DIV r/m	Ділення акумулятора на беззнакове число (ділення на нуль зумовлює переривання *INT* 0)	$\|AX\| : \|reg8\| \to AL$ (остача *AH*)	2	80 – 90
		$\|DX, AX\| : reg16 \to AX$ (остача *DX*)	2	144 – 162
		$\|AX\| : \|mem8\| \to AL$ (остача *AH*)	2–4*1	86–96 + + *EA*
		$\|DX, AX\| : \|mem16\| \to$ $\to AX$ (остача *DX*)	2–4*1	150–168 + EA
IDIV r/m	Ділення акумулятора на ціле число (8- або 16-розрядне). Ділення на нуль зумовлює переривання *INT* 0	$AX : reg8 \to AL$ (остача *AH*)	2	101–112
		$(DX, AX) : reg16 \to AX$ (остача *DX*)	2	165–184
		$AX : mem8 \to AL$ (остача *AH*)	2–4*1	144–168+ + *EA*
		$DX, AX : mem16 \to$ $\to AX$ (остача *DX*)	2–4*1	166–190+ + *EA*
AAD	Корекція перед діленням розпакованих двійково-десяткових чисел		2	60
CBW	Перетворення байта *AL* на слово *AX* (повторення вмісту знакового розряду (*AL*7) регістра *AL* в усіх розрядах регістра *AH*)	$AH \leftarrow (AL7)$	1	2
CWD	Перетворення слова *AX* на подвійне слово *DX, AX* (повторення вмісту знакового розряду (*AX*15) регістра *AX* в усіх розрядах регістра *DX*)	$DX \leftarrow AX15)$	1	5
	Логічні команди			
NOT r/m	Інверсія (інверсія всіх бітів операнда)	$reg \leftarrow \overline{reg}$	2	3
		$mem \leftarrow \overline{mem}$	2–4*1	16 + *EA*
AND r/m, r/m/i	Логічне І двох операндів	$reg \leftarrow reg \wedge reg$	2	3
		$mem \leftarrow mem \wedge reg$	2–4*1	16 + *EA*
		$reg \leftarrow reg \wedge mem$	2–4*1	9 + *EA*
		$reg8 \leftarrow reg8 \wedge immed$	3	4
		$reg16 \leftarrow reg16 \wedge immed$	4	4
		$mem8 \leftarrow mem8 \wedge immed$	3–5*2	17 + *EA*
		$mem16 \leftarrow mem16 \wedge$ $\wedge immed$	4–6*3	17 + *EA*

Продовж. табл. 2.11

Мнемокод команди	Опис команди	Алгоритм команди	Кіль-кість байт	Кіль-кість тактів
		AL ← AL ∧ *immed*	2	4
		AX ← AX ∧ *immed*	3	4
OR r/m, r/m/i	ЛОГІЧНЕ АБО двох операндів	*reg* ← *reg* ∨ *reg*	2	3
		mem ← *mem* ∨ *reg*	2–4*1	16 + *EA*
		reg ← *reg* ∨ *mem*	2–4*1	9 + *EA*
		*reg*8 ← *reg*8 ∨ *immed*	3	4
		*reg*16 ← *reg*16 ∨ *immed*	4	4
		*mem*8 ← *mem*8 ∨ *immed*	3–5*2	17 + *EA*
		*mem*16 ← *mem*16 ∨ ∨ *immed*	4–6*3	17 + *EA*
		AL ← *AL* ∨ *immed*	2	4
		AX ← *AX* ∨ *immed*	3	4
XOR r/m, r/m/i	ВИКЛЮЧАЛЬНЕ АБО двох операндів	*reg* ← *reg* ⊕ *reg*	2	3
		mem ← *mem* ⊕ *reg*	2–4*1	16 + *EA*
		reg ← *reg* ⊕ *mem*	2–4*1	9 + *EA*
		*reg*8 ← *reg*8 ⊕ *immed*	3	4
		*reg*16 ← *reg*16 ⊕ ⊕ *immed*	4	4
		*mem*8 ← *mem8* ⊕ ⊕ *immed*	3–5*2	17 + *EA*
		*mem*16 ← *mem16* ⊕ ⊕ *immed*	4–6*3	17 + *EA*
		AL ← *AL* ⊕ *immed*	2	4
		AX ← *AX* ⊕ *immed*	3	4
TEST r/m, r/m/i	Перевірка (логічне І без запису результату і встановлення прапорців відповідно до результату)	*F* ⇐ *reg* ∧ *reg*	2	3
		F ⇐ *mem* ∧ *reg*	2–4*1	9 + *EA*
		F ⇐ *reg* ∧ *mem*	2–4*1	9 + *EA*
		F ⇐ *reg*8 ∧ *immed*	3	4
		F ⇐ *reg*16 ∧ *immed*	4	4
		F ⇐ *mem*8 ∧ *immed*	3–5*2	10 + *EA*
		F ⇐ *mem*16 ∧ *immed*	4–6*3	10 + *EA*
		F ⇐ *AL* ∧ *immed*	2	4
		F ⇐ *AX* ∧ *immed*	3	4
	Команди зсуву			
RCL/RCR r, 1	Циклічний зсув ліворуч-праворуч через біт *CF*: на одну позицію		2	2
RCL/RCR m, 1			2–4*1	15 + *EA*
RCL/RCR r, CL	на *CL* позицій	ліворуч	2	8 + *4CL*
RCL/RCR m, CL		праворуч	2–4*1	20+ + *EA* + + *4CL*

Мнемокод команди	Опис команди	Алгоритм команди	Кіль-кість байт	Кіль-кість тактів
ROL/ROR r,1 *ROL/ROR m,1* *ROL/ROR r, CL* *ROL/ROR m, CL*	Циклічний зсув ліворуч-праворуч: на одну позицію на *CL* позицій *SAL/SAR*  ліворуч  праворуч		2 2–4*1 2 2–4*1	2 15 + *EA* 8 + 4*CL* 20 + + *EA* + + 4*CL*
SAL/SAR r, 1 *SAL/SAR m,1* *SAL/SAR r, CL* *SAL/SAR m, CL*	Зсув арифметичний ліворуч-праворуч: на одну позицію на *CL* позицій  0 ліворуч  праворуч Під час зсуву праворуч значення біта у старшому розряді залишається незмінним		2 2–4*1 2 2–4*1	2 15 + *EA* 8 + 4*CL* 20 + + *EA* + + 4*CL*
SHR r, 1 *SHR m, 1* *SHR r, CL* *SHR m, CL*	Зсув логічний праворуч[4]: на одну позицію на *CL* позицій 0  		2 2–4*1 2 2–4*1	2 15 + *EA* 8 + 4*CL* 20 + + *EA* + + 4*CL*
	Рядкові команди			
REP	Префікс повторення рядкових операцій триває доти, поки *CX* ≠ 0 (*CX* декрементується після виконання кожної рядкової операції)		1	
REPE (REPZ)	Префікс умовного повторення – повторення за *ZF* = 1 або за *CX* ≠ 0		1	
REPNE (REPNZ)	Префікс умовного повторення – повторення за *ZF* = 0 або за *CX* ≠ 0		1	
MOVSB	Копіювання байта з комірки пам'яті з адресою *DS*: [*SI*] у комірку *ES*:[*DI*]	*ES*: [*DI*]← ← *DS*: [*SI*] *SI* ← *SI* ± 1*7; *DI* ← *DI* ± 1*7	1	18*5 9 + +17*CX*6

Продовж. табл. 2.11

Мнемокод команди	Опис команди	Алгоритм команди	Кількість байт	Кількість тактів
MOVSW	Копіювання слова з *DS*: [*SI*] у *ES*: [*DI*]	*ES*: [*DI*] ← ← *DS*: [*SI*] *SI* ← *SI*±2[*7]; *DI* ← *DI*±2[*7]	1	18[*5] 9 + +17*CX*[*6]
LODSB	Копіювання байта з *DS*: [*SI*] в *AL*	*AL* ← *DS*: [*SI*] *S I*← *SI*±1[*7]	1	11[*5] 9 + +10 *CX*[*6]
LODSW	Копіювання слова з *DS*: [*SI*] в *AX*	*AX* ← *DS*: [*SI*] *SI* ← *SI*±2[*7]	1	11[*5] 9 + +10 *CX*[*6]
STOSB	Запис байта з *AL* в *ES*: [*DI*]	*ES*: [*DI*] ← *AL* *DI* ← *DI* ± 1[*7]	1	11[*5] 9 + +10 *CX*[*6]
STOSW	Запис слова з *AX* в *ES*: [*DI*]	*ES*: [*DI*] ← *AX* *DI* ← *DI* ± 2[*7]	1	11[*5] 9 + +10 *CX*[*6]
CMPSB	Порівняння байтів *DS*: [*SI*] і *ES*: [*DI*] із записом результату порівняння у регістр прапорців	*F* ⇐ *ES*: [*DI*] − − *DS*: [*SI*] *SI* ← ← *SI* ± 1[*7]; *DI* ← *DI* ± 1[*7]	1	22[*5] 9 + +22 *CX*[*6]
CMPSW	Порівняння слів *DS*: [*SI*] і *ES*: [*DI*] із записом результату порівняння у регістр прапорців	*F* ⇐ *ES*: [*DI*] − − *DS*: [*SI*] *SI* ← ← *SI* ± 2[*7]; *DI* ← *DI*± 2[*7]	1	22[*5] 9 + +22 *CX*[*6]
SCASB	Порівняння байта *DS*: [*SI*] і *AL* із записом результату порівняння у регістр прапорців	*F* ⇐ [*DS*: *SI*] −*AL*; *SI* ← *SI* ± 1[*7]	1	15[*5] 9 + +15 *CX*[*6]
SCASW	Порівняння слова [*DS*: *SI*] і *AX* із записом результату порівняння у регістр прапорців	*F* ⇐ [*DS*: *SI*] − *AX*; *SI* ← *SI* ± 2[*7]	1	15[*5] 9 + +15 *CX*[*6]
КОМАНДИ ПЕРЕДАВАННЯ КЕРУВАННЯ				
*JMP target*16	Внутрішньосегментний безумовний перехід до цільової адресси *target* (перехід у межах сегмента завдовжки 64 Кбайт)	*IP* ← *IP* + *disp*16	3	15
*JMP NEAR target*8		*IP* ← *IP* + *disp*8	2	15
JMP reg		*IP* ← *IP* + *reg*	2	2
JMP mem		*IP* ← *IP* + *mem*	2–4[*1]	18 + *EA*

Продовж. табл. 2.11

Мнемокод команди	Опис команди	Алгоритм команди	Кількість байт	Кількість тактів
JMP FAR target *JMP FAR mem*	Міжсегментний безумовний перехід до цільової адреси *target* (перехід у межах усієї ємності пам'яті 1 Мбайт)	*IP* ← *offset target*, *CS* ← *seg target* *IP* ← [*mem*] *IP* ← [*mem*+2]	5 2–4*1	15 24 + *EA*
JCX target	Перехід, якщо *CX* = 0		2	18/6*8
LOOP target	Цикл: *CX* ← *CX* − 1 і перехід, якщо *CX* ≠ 0		2	16/4*8
LOOPE (LOOPZ) target	*CX* ← *CX* − 1 і перехід, якщо *CX* ≠ 0 і *ZF* = 1		2	18/6*8
LOOPNE (LOOPNZ) target	*CX* ← *CX* − 1 і перехід, якщо *CX* ≠ 0 і *ZF* = 0		2	19/5*8
JA (JNBE) target	Перехід, якщо перший беззнаковий операнд більше, ніж другий (*CF* = *ZF* = 0)		2	16/4*8
JAE (JNB) target	Перехід, якщо перше беззнакове число не менше, ніж друге (*CF* = 0)		2	16/4*8
JB (JC) target	Перехід, якщо перше беззнакове число менше, ніж друге (*CF* = 1)		2	16/4*8
JE (JZ) target	Перехід, якщо числа дорівнюють (*ZF* = 1)		2	16/4*8
JG (JNLE) target	Перехід, якщо перше більше за друге знакове число (*SF* = (*ZF*&*OF*))		2	16/4*8
JGE (JNL) target	Перехід, якщо перше знакове число більше або дорівнює другому (*SF* = *OF*)		2	16/4*8
JL (JNGE) target	Перехід, якщо перше знакове число менше за друге (*SF* ≠ *OF*)		2	16/4*8
JLE (JNG) target	Перехід, якщо перше знакове число менше або дорівнює другому (*SF* ≠ *OF* або *ZF* = 0)		2	16/4*8
JNC (JAE/JNB) target	Перехід, якщо перенесення немає (*CF* = 0)		2	16/4*8
JNE (JNZ) target	Перехід, якщо числа не рівні (*ZF* = 0)		2	16/4*8
JNO target	Перехід, якщо немає переповнення (*OF* = 0)		2	16/4*8

Продовж. табл. 2.11

Мнемокод команди	Опис команди	Алгоритм команди	Кількість байт	Кількість тактів
JNP (JPO) target	Перехід, якщо паритет непарний (*PF* = 0)		2	16/4*8
JNS target	Перехід, якщо позитивний результат (*SF* = 0)		2	16/4*8
JO target	Перехід, якщо є переповнення (*OF* = 1)		2	16/4*8
JP (JPE) target	Перехід, якщо паритет парний (*PF* = 1)		2	16/4*8
JS target	Перехід, якщо негативний результат (*SF* = 1)		2	16/4*8
CALL NEAR target	Внутрішньосегментний виклик процедури (виклик у межах сегмента завдовжки 64 Кбайт)	*SP* ← *SP* − 2 [*SS*:*SP*] ← *IP*; *IP* ← *target*	3	19
CALL NEAR reg		*IP* ← *reg*	2	16
CALL NEAR mem		*IP* ← *mem*	2–4*1	21 + *EA*
CALL FAR target	Міжсегментний виклик процедури (виклик у межах усієї ємності пам'яті 1 Мбайт)	*SP* ← *SP* − 2 [*SS*:*SP*] ← *CS* *SP* ← *SP* − 2 [SS:SP] ← IP; *CS*, *IP* ← *target*	5	28
CALL FAR mem		*IP* ← [*mem*]; *CS* ← [*mem* + 2]	2–4*1	37 + *EA*
RET *RET NEAR*	Повернення з внутрішньосегментної процедури.	*IP* ← [*SS*:*SP*]; *SP* ← *SP* + 2	1	8
RET (n) *RET NEAR (n)*	Необов'язковий параметр *n* задає корекцію значення вказівника стеку	*IP* ← *[SS:SP];* *SP* ←*SP* + *n*	3	12
RET FAR	Повернення з міжсегментної процедури	*IP* ← [*SS*:*SP*]; *SP* ←*SP* + 2	1	18
RET FAR (n)	Необов'язковий параметр *n* задає корекцію значення вказівника стеку	*CS* ← [*SS*:*SP*]; *SP* ← *SP* + 2	3	17
	Команди переривань			
INT n	Виконання програмного переривання	*SP* ← *SP* − 2 [*SS*:*SP*] ← F	2	51

Мнемокод команди	Опис команди	Алгоритм команди	Кількість байт	Кількість тактів
		$SP \leftarrow SP - 2$ $[SS{:}SP] \leftarrow CS$ $SP \leftarrow SP - 2$ $[SS{:}SP] \leftarrow IP$		
INT 3	Виконання програмного переривання 3		1	52
INTO	Виконання програмного переривання 4, якщо прапорець $OF = 1$		1	53/4[*9]
IRET	Повернення з переривання	$IP \leftarrow [SS{:}SP]$; $SP \leftarrow SP+2$ $CS \leftarrow [SS{:}SP]$; $SP \leftarrow SP+2$; $F \leftarrow [SS{:}SP]$; $SP \leftarrow SP+2$	1	24
КОМАНДИ КЕРУВАННЯ СТАНОМ МП				
CLC	Скидання прапорця перенесення	$CF \leftarrow 0$	1	2
CMC	Інверсія прапорця перенесення	$CF \leftarrow \overline{CF}$	1	2
STC	Установлення прапорця перенесення	$CF \leftarrow 1$	1	2
CLD	Скидання прапорця напряму	$DF \leftarrow 0$	1	2
STD	Установлення прапорця напряму	$DF \leftarrow 1$	1	2
CLI	Заборона маскованих апаратних переривань	$IF \leftarrow 0$	1	2
STI	Дозвіл маскованих апаратних переривань	$IF \leftarrow 1$	1	2
HLT	Зупинка процесора		1	2
WAIT	Очікування сигналу на лінії *TEST*		1	3
ESC msk/mem	Передавання коду команди *msk* або коду та операнда *mem* арифметичному співпроцесору		2 2–4[*1]	2 8 + *EA*

Продовж. табл. 2.11

Мнемокод команди	Опис команди	Алгоритм команди	Кількість байт	Кількість тактів
LOCK	Префікс блокування шини на час виконання наступної інструкції у максимальному режимі		1	2
NOP	Операцій немає		1	2

Примітки:

*[1]Команда займає два байти, якщо під час адресації комірки пам'яті не використовується зміщення, тобто *disp* = 0, наприклад, позначення комірки пам'яті *mem* відповідає позначенню [*SI*]; команда займає три байти, якщо використовується 8-розрядне зміщення *disp*8, наприклад, [*SI* + 25*H*]; команда займає чотири байти, якщо зміщення16-розрядне – *disp*16, наприклад, [*SI* + 1000*H*].

*[2]Команда займає три байти за *disp* = 0, чотири байти за *disp*8 і п'ять байт при *disp*16.

*[3]Команда займає чотири байти за *disp* = 0, п'ять байт за *disp*8 і шість байт за *disp*16.

*[4]Логічний зсув логічний ліворуч збігається з арифметичним зсувом ліворуч. Допускається замість позначення *SAL* використовувати позначення *SHL*.

*[5]Наведено тривалість виконання рядкової команди без префікса повторення.

*[6]Наведено тривалість виконання рядкової команди із префіксом повторення. У регістрі *CX* записано кількість повторень.

*[7]За встановленого прапорця напряму (*DF* = 1) – це операція додавання або віднімання.

*[8]*m*/*n* – під час виконання переходу команда виконується за *m* або за *n* тактів.

*[9]За встановленого прапорця (*OF* = 1) команда виконується за 53 або за чотири такти.

Вплив команд на значення прапорців ілюструє табл. 2.13, в якій позначено: «+» — команда впливає на прапорець; «–» — не впливає; «1» — встановлює прапорець у стан логічної одиниці; «0» — скидає прапорець у стан логічного нуля; «?» — стан невизначений (залежить від конкретних значень операндів).

Приклади виконання команд. Розглянемо приклади виконання команд передавання інформації. До цієї групи команд належать команди пересилання даних з регістрів у регістри, з регістрів у пам'ять, з пам'яті у регістри, з пам'яті до пам'яті, в тому числі у стек та зі стеку.

Таблиця 2.12. **Обчислення ефективної адреси *EA***

Адресація	Спосіб обчислення	Кількість тактів
Пряма	[*disp*]	6
Непряма	[*BX*], [*BP*], [*SI*], [*DI*]	5
Базова або індексна	[*BX* + *disp*], [*BP* + *disp*], [*SI* + *disp*], [*DI* + *disp*]	9
Базово-індексна без зміщення	[*BP* + *DI*], [*BX* + *SI*], [*BP* + *SI*], [*BX* + *DI*]	7 8
Базово-індексна зі зміщенням	[*BP* + *DI* + *disp*], [*BX* + *SI* + *disp*], [*BP* + *SI* + *disp*], [*BX* + *DI* + *disp*]	11 11 12 12

Приклад 2.12. Передати дані з регістра *CL* у регістр *BL*. До виконання команди регістр *BL* містить число 101011111, а *CL* — 00001111.

За командою

$$MOV\ BL,CL\ ; \qquad BL \leftarrow CL$$

(*MOVe* — переслати) вміст регістра *CL* пересилається у регістр *BL*. Після виконання команди вміст регістра *BL* дорівнює 00001111, вміст *CL* не змінюється, тобто в МП вміст двох регістрів стає однаковим:

$$BL = 00001111,\ CL = 00001111.$$

Приклад 2.13. Переслати вміст комірки пам'яті *DS*:[100*EH*] у регістр *CX*.

За командою

$$MOV\ CX,\ [00EH]\ ; \qquad CX \leftarrow DS{:}[100EH]$$

вміст 16-розрядної комірки пам'яті з адресою *DS*:[100*EH*] пересилається у 16-розрядний регістр *CX*. Нехай до виконання команди вміст регістра *CX* дорівнював 1234*H*, тобто 0001001000110100_2, а в пам'яті за вказаною адресою знаходиться слово 5678*H*, причому молодший байт слова 78*H* розташований за адресою *DS*:[100*EH*], старший байт 56*H* — за адресою *DS*:[100*FH*]. Після виконання команди молодший байт слова в пам'яті перепишеться у молодшу частину регістра *CX*, тобто на *CL*, а старший байт — у старшу частину, тобто на *CH*, унаслідок чого вміст *CX* буде 5678*H*, а інформація у пам'яті залишиться незмінною (рис. 2.20).

Зазначимо, що у двохоперандних командах типу

$$MOV\ dest,\ src$$

один з операндів має бути вмістом регістра:

$$MOV\ r1,\ r2\ ;\ r1 \leftarrow r2$$
$$MOV\ m,\ r\ ;\ m \leftarrow r$$
$$MOV\ r,\ m\ ;\ r \leftarrow m$$

Таблиця 2.13. **Встановлення прапорців**

Операція	Команди	Прапорці								
		OF	*CF*	*AF*	*SF*	*ZF*	*PF*	*DF*	*IF*	*TF*
Додавання, віднімання	*ADD, ADC, SUB, SBB*	+	+	+	+	+	+	–	–	–
	CMP, NEG, CMPS, SCAS	+	+	+	+	+	+	–	–	–
	INC, DEC	+	–	+	+	+	+	–	–	–
Множення	*MUL, IMUL*	+	+	?	?	?	?	–	–	–
Ділення	*DIV, IDIV*	?	?	?	?	?	?	–	–	–
Десяткова корекція	*DAA, DAS,*	?	+	+	+	+	+	–	–	–
	AAA, AAS	?	+	+	?	?	?	–	–	–
	AAM, AAD	?	?	?	+	+	+	–	–	–
Логічні команди	*AND, OR, XOR, TEST*	0	0	?	+	+	+	–	–	–
Зсув	*RCL, RCR,*	+	+	?	–	–	–	–	–	–
	ROL, ROR dest – dest, CL	?	+	?	–	–	–	–	–	–
	SHL, SHR dest	+	+	?	+	+	+	–	–	–
	–dest, CL	?	+	?	+	+	+	–	–	–
	SAR	0	+	?	+	+	+	–	–	–
Відновлення прапорців	*POPF, IRET*	+	+	+	+	+	+	+	+	+
	SAHF	–	+	+	+	+	+	–	–	–
Переривання	*INT, INTO*	–	–	–	–	–	–	–	0	0
Керування прапорцями	*STC*	–	1	–	–	–	–	–	–	–
	CLC	–	0	–	–	–	–	–	–	–
	CMC	–	$\overline{CF}$	–	–	–	–	–	–	–
	STD	–	–	–	–	–	–	1	–	–
	CLD	–	–	–	–	–	–	0	–	–
	STI	–	–	–	–	–	–	–	1	–
	CLI	–	–	–	–	–	–	–	0	–

Приклад 2.14. Завантажити сегментний регістр *DS* початковим значенням 4000*H*.

Оскільки команди *MOV sr, immed* не існує, то для того щоб завантажити в сегментний регістр будь-яке значення, треба записати його у РЗП, а потім передати його в сегментний регістр, тобто використати дві команди:

$$MOV\ AX,\ 4000H\ ;\ AX \leftarrow 4000H$$

$$MOV\ DS,\ AX\ ;\ DS \leftarrow AX.$$

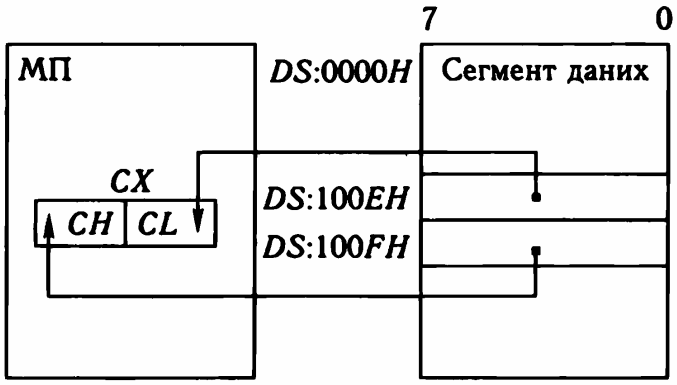


Рис. 2.20. Виконання команди *MOV* [100*EH*], *CX*

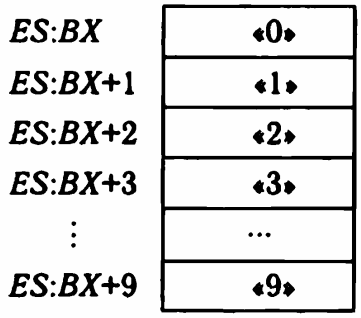


Рис. 2.21. Розміщення семисегментних кодів у пам'яті

Приклад 2.15. Здійснити обмін вмісту двох регістрів — *BL* і *CL*. За командою

$$XCNG\ BL,CL\ ;\ BL \longleftrightarrow CL$$

(*eXChaNGe* — обмін) вміст регістрів *BL* і *CL* обмінюється місцями.

Нехай значення регістрів *BL* = 10101111, *CL* = 00001111. Після виконання команди вміст регістрів буде

$$BL = 00001111;$$

$$CL = 10101111.$$

Зазначимо, що під час виконання цієї команди, крім регістрів, які беруть участь у команді, використовуються буферні регістри МП, що зберігають проміжні дані.

Приклад 2.16. Нехай в таблиці з початковою адресою *ES : BX* послідовно розташовано байти семисегментного коду цифр від 0 до 9 (рис. 2.21). Замінити значення вмісту регістра *AL* на відповідний семисегментний код.

Для вирішення завдання скористаємося командою

$$XLAT\ ; \qquad AL \leftarrow ES{:}[BX + AL],$$

де *XLAT* (*indeX Load Accumulator from Table*) — індексне (за непрямою адресацію) завантаження акумулятора даними (див. рис. 2.21), що замінює вміст *AL* на вміст комірки пам'яті, розташованої у сегменті *ES* зі зміщенням *BX* + *AL*.

Цю команду використовують для перекодування символу, який знаходиться в *AL*, на байт з таблиці перекодування. Для коректної роботи команди треба, щоб таблиця була розташована за певною адресою, а саме за початковою адресою *ES:BX*. Довжина таблиці не повинна перевищувати 256 байт.

Якщо в регістрі *AL* до виконання команди знаходиться, наприклад, число 3, то після виконання в *AL* буде семисегментний код числа 3.

Команду *XLAT* доцільно використовувати для заміни аргументу значенням функції, а значення функцій заздалегідь записувати у таблиці.

Приклад 2.17. Запам'ятати в стеку вміст 16-розрядного регістра загального призначення.

За командою

$$PUSH\ AX \quad ; SP \leftarrow SP - 2, AX \leftarrow SS : [SP],$$

(*PUSH* — занести) вміст покажчика стеку *SP* зменшується на два, тобто $SP \leftarrow SP - 2$, для адресування верхньої вільної комірки стеку. Після цього вміст регістра *AX* запам'ятовується у 16-розрядній комірці стеку з адресою *SS : SP*. Виконання команди наведено на рис. 2.22.

Команди запису у стек та зчитуванням зі стеку можуть оперувати лише з 16-розрядними операндами.

Операції зі стеком є більш швидкодіючими порівняно зі зверненнями до пам'яті з довільною вибіркою завдяки тому, що адреса комірки стеку формується автоматично. Програміст має лише ініціювати вершину стеку на початку програми, тобто записати в регістри *SS* і *SP* початкові значення:

$$MOV\ AX, 7000H \quad ; AX \leftarrow 7000H$$
$$MOV\ SS, AX \quad ; SS \leftarrow AX$$
$$MOV\ SP, 4000H \quad ; SP \leftarrow 4000H$$

Початковою адресою стеку в цьому прикладі є 7000*H*:4000*H*.

Розглянемо приклади виконання арифметичних, логічних команд та команд зсувів.

Приклад 2.18. Додати вміст двох регістрів: *CL* і *DL*. До виконання команди вміст регістрів: *CL* = 10011100, *DL* = 11000101.

За командою

$$ADD\ CL,DL \quad ; CL \leftarrow CL + DL$$

(*ADD* — додати) додаються два операнди; при цьому результат записується на місце першого операнда. Після виконання команди у регістрі *CL* буде значення 01100001:

```
рядок перенесень  1  11
                   10011100
                  +
                   11000101
                  ---------
                   01100001
```

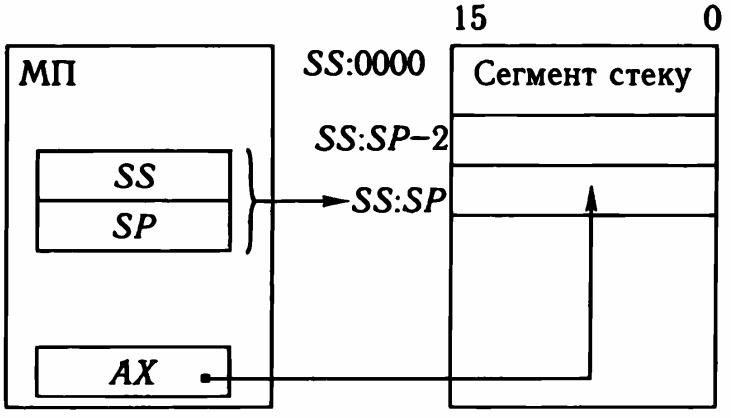


Рис. 2.22. Виконання команди *PUSH AX*

Усі арифметичні команди впливають на встановлення прапорців. Так, після виконання цієї команди прапорці встановлюють:

у стан одиниці: *AF* — прапорець перенесення з розряду *D*3 в розряд *D*4; *CF* — прапорець перенесення з розряду *D*7; *OF* — прапорець переповнення (в результаті додавання двох знакових від'ємних чисел отримано позитивний результат);

у стан нуля: *ZF* — прапорець нуля (результат ненульовий); *PF* — прапорець паритету (число одиниць у результаті непарне); *SF* — прапорець знака (знаковий розряд у результаті дорівнює нулю).

Приклад 2.19. Виконати додавання чотирибайтових операндів, які містяться у сусідніх комірках пам'яті з початковими адресами *DS*:1000H і *DS* : 2000*H*. Результат запам'ятати у регістрах *CX*, *DX*.

Для виконання цього завдання треба переслати вміст молодшого слова першого доданка у регістр *CX* та скласти з молодшим словом другого доданка за допомогою команди *ADD*:

MOV CX,[1000*H*] ; *CX* ← *DS*:[1000*H*]

ADD CX, [2000*H*] ; *CX* ← *CX* + *DS*:[2000*H*]

Для того щоб врахувати перенесення, яке може виникнути під час додавання молодших слів, треба виконати додавання наступних слів операндів за допомогою команди *ADC*, що передбачає додавання значення біта *CF* до отриманої суми. Якщо *CF* = 0, сума не змінюється.

Отже, треба переслати вміст старшого слова першого доданка у регістр *DX* та скласти зі старшим словом другого доданка:

MOV DX,[1002*H*] ; *DX* ← *DS*:[1002*H*]

ADC DX, [2002*H*] ; *DX* ← *DX*+ *DS*:[2002*H*] + біт *CF*

Якщо слова займають у пам'яті дві сусідні комірки, для адресації старшого слова доданка адресу слід збільшити на два.

Алгоритм команди додавання з урахуванням перенесення *ADC DX*, [2002*H*] (*ADC — Add with Carry* — додати з перенесенням) наведено на рис. 2.23.

Додавання наступних байтів багатобайтових операндів виконується так: молодші байти операндів додаються за командою *ADD*, а всі інші байти — за командою *ADC*.

Приклад 2.20. Виконати віднімання від вмісту регістра *AX* числа 5000*H*.

За командою

*SUB AX,*5000*H* ; *AX* ← *AX* — 5000*H*

(*SUB — SUBtract* — відняти) різниця *AX*— 5000*H* пересилається в акумулятор *AX*.

Приклад 2.21. Виконати віднімання вмісту комірки пам'яті з адресою *DS*:1000*H* та числа 35*H* з урахуванням позики від попередньої операції.

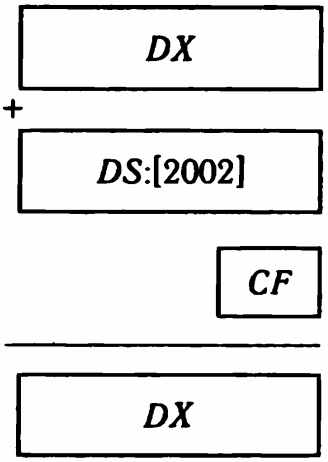


Рис. 2.23. Виконання команди *ADC DX*, [2002*H*]

Для того щоб врахувати позику, треба виконати віднімання за допомогою команди *SBB*, яка передбачає віднімання біта *CF* від отриманої різниці [*DS*:1000*H*] – 35*H*. Якщо *CF* = 0, то результат віднімання залишиться незмінним:

SBB [1000*H*],35*H* ; [*DS*:1000*H*] ← [*DS*:1000*H*] – 35*H*-біт *CF*

Команду *SBB* використовують для віднімання багатобайтових операндів з урахуванням позики з попередніх операцій. Віднімання багатобайтових операндів здійснюється так: молодші байти віднімаються за командою *SUB*, а всі інші — за командою *SBB*.

Приклад 2.22. Визначити додатковий код операнда, який міститься у регістрі BX.

За командою

$$NEG\ BX \quad ;\ BX \leftarrow \overline{BX} + 1$$

(*NEG — NEGative*) виконується переведення числа у додатковий код, тобто спочатку здійснюється його побітова інверсія, а потім додавання одиниці.

Приклад 2.23. Порівняти значення вмісту акумулятора і регістра *CX*.

За командою

$$CMP\ AX,\ CX \quad ;\ AX - CX \Rightarrow F$$

(*CoMPare* — порівняти) від вмісту *AX* віднімається вміст *CX*. При цьому різниця ніде ні фіксується, але згідно з нею встановлюються прапорці. Так, прапорець нуля *ZF* установлюється у стан логічної одиниці, якщо числа тотожні. Якщо значення *AX* за модулем менше ніж *CX*, встановлюється прапорець позики *CF*.

Приклад 2.24. Знайти добуток двох беззнакових 8-розрядних операндів, які знаходяться у регістрах *DL* і *AL*.

У команді множення *MUL* (*MULtiplay* — перемножити) беззнакових чисел вказується лише один операнд — *DL*, оскільки другий множник за замовчуванням знаходиться в акумуляторі. Результат множення байтів розміщується у 16-розрядному регістрі *AX*:

$$MUL\ DL \quad ;\ AL \times DL \rightarrow AX.$$

Приклад 2.25. Знайти добуток двох беззнакових 16-розрядних операндів, які знаходяться у регістрах *CX* і *AX*.

Під час множення двобайтових операндів у мнемоніці команди *MUL* вказується лише один операнд *CX*. Другий операнд за замовчуванням знаходиться в акумуляторі *AX*. Молодша частина добутку зберігається в *AX*, старша — в *DX*. Результат розміщується у регістрах *AX* і *DX*:

$$MUL\ CX \quad ;\ AX \times CX \rightarrow (DX,\ AX).$$

Приклад 2.26. Знайти результат ділення вмісту акумулятора AX на вміст регістра *CL*.

За командою

$$DIV\ CL \quad ;\ AX:\ CL \rightarrow AL,\ \text{остача} \rightarrow AH$$

(*DIVide* — поділити) вміст *AX* ділиться на вміст *CL*, результат записується в *AL*, а остача від ділення — в *AH*.

Приклад 2.27. Виконати команду ділення подвійного слова, що зберігається у регістрах *DX* і *AX*, на слово у регістрі *CX*.

Під час ділення подвійного слова на слово у мнемоніці команди *DIV* вказується лише дільник, який знаходиться у регістрі *CX*. Молодша частина діленого за замовчуванням знаходиться у регістрі *AX*, старша частина — у регістрі *DX*. Результат розміщується у регістрах *AX* і *DX*:

$$DIV\ CX \quad ;(DX, AX) : CX \rightarrow AX, \text{ остача} \rightarrow \text{DX}.$$

Приклад 2.28. Виконати команду десяткової корекції результату додавання двох упакованих двійково-десяткових чисел, які знаходяться у регістрах $AL = 25H$, $BL = 37H$. Команду додавання записати так, щоб після її виконання результат був розміщений у регістрі *AL*.

Команда

$$DAA$$

(*Decimal correction of Accumulator at Addition* — десяткова корекція акумулятора під час додавання) використовується після команди додавання і перетворює результат додавання двійково-десяткових чисел на двійково-десяткове число. Корекція полягає в узгодженні перенесень у процесі додавання десяткових та шістнадцяткових чисел. Така корекція необхідна, оскільки за порозрядного додавання десяткових чисел перенесення виконується, якщо результат перевищує 9, а за порозрядного додавання шістнадцяткових чисел, якщо результат перевищує $F = 15_{10}$. Алгоритм десяткової корекції такий:

- якщо прапорець $AF = 1$ або молодша тетрада $AL > 9$, то $AL \leftarrow AL + 06, AF \leftarrow 1$;
- якщо прапорець $CF = 1$ або старша тетрада $AL > 9$, то $AL \leftarrow AL + 60H, CF \leftarrow 1$.

Після виконання команди

$$ADD\ AL,BL$$

вміст регістра *AL* визначається так:

$$\begin{array}{r} 00100101 \\ + \\ \underline{00110111} \\ 01011100 \end{array} = 5CH.$$

Це число містить літери, тобто не є двійково-десятковим числом, тому після команди *ADD* виконується команда *DAA*, в результаті виконання якої результат додавання змінюється. Значення молодшої тетради результату більше ніж 9. Згідно з алгоритмом

$$\begin{array}{r} 01011100 \\ + \\ \underline{110} \\ 01100010_{2-10} \end{array} = 62.$$

Дійсно, 25 + 37 = 62. Результат додавання 62 є двійково-десятковим числом.

Приклад 2.29. Здійснити порозрядну операцію ЛОГІЧНЕ АБО над числами, які містяться у регістрах *BL*=11011010 та *CL*=10001001.

Команда

$$OR\ BL,\ CL \quad ;\ BL \leftarrow BL\ v\ CL$$

(*OR* — або) виконує операцію ЛОГІЧНЕ АБО, результат якої розміщується на місці першого операнда. Після виконання команди $BL = 11011011$.

Так само порозрядно діють усі логічні операції: *AND, XOR, NOT.*

Логічні команди діють на прапорці так: прапорці *CF* і *OF* встановлюються у стан логічного нуля, прапорці *ZF, SF, PF* — відповідно до результату (див. табл. 2.10), а значення прапорця *AF* не визначено. Після виконання команди прапорці набувають таких значень: $ZF = 0$, $SF = 1$, $PF = 1$.

Приклад 2.30. Установити прапорці відповідно до результату операції ЛОГІЧНЕ І над числом $38H$, яке міститься у регістрі *CL*, та числом $35H$.

За командою

$$TEST\ CL,35H \quad ;\ CL \wedge 35H \Rightarrow F$$

(*TEST* — перевірка) здійснюється кон'юнкція операндів, за результатом встановлюються прапорці, але результат операції не фіксується. Після виконання команди

$$\begin{array}{r} 00111000 \\ \wedge \\ \underline{00110101} \\ 00110000 \end{array}$$

прапорці набувають значень відповідно до результату операції: $ZF = 0$, $PF = 1$, $SF = 0$, $CF = 0$, $OF = 0$, стан прапорця *AF* не визначений.

Приклад 2.31. Виконати циклічний зсув вмісту регістра *BX* ліворуч на один розряд.

За командою

$$ROL\ BX,\ 1$$

або

$$ROL\ BX,\ CL$$

(*ROL — ROtate shift Left* — циклічний зсув ліворуч) здійснюється циклічний зсув вмісту регістра *BX* ліворуч відповідно на один або *CL* розрядів (рис. 2.24). За результатами операції змінюються значення прапорців *CF, SF, ZF, PF*. Команда зсуву на один розряд змінює також прапорець *OF* — він установлюється у стан «1» за відповідної зміни знакового розряду.

Розглянемо приклади виконання групи *рядкових команд.* Рядкові команди оперують з масивами даних рядками. За замовчуванням один з масивів (джерело інформації) знаходиться в сегменті даних *DS* з початковою адресою, яка визначається вмістом регістра *SI*, а другий масив (приймач інформації) — у додатковому сегменті даних *ES* з початковою адресою, яка визначається вмістом регістра *DI*. Елементами рядків є слова або байти. Напрям оброблення інформації у рядках

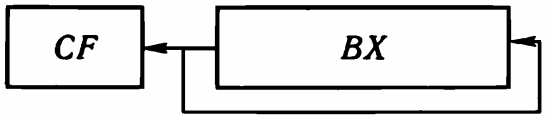


Рис. 2.24. Виконання команди *ROL BX,1*

визначається прапорцем напряму *DF*. За *DF* = 0 оброблення інформації починається з молодших адрес, за *DF* = 1 — зі старших адрес.

Приклад 2.32. Передати вміст 8-розрядної комірки пам'яті з адресою *DS*:*SI* у комірку пам'яті з адресою *ES*:*DI*.

Для виконання цього завдання використовується команда *MOVSB* (*MOVe Serial Byte* — перемістити послідовність байтів), яка пересилає байт або послідовність байтів з однієї області пам'яті в іншу. Якщо у мнемоніці команди використовується префікс *REP*, то пересилається послідовність байтів, при цьому кількість байтів у послідовності задається вмістом регістра *CX*; якщо у мнемоніці команди префікса *REP* немає, то пересилається вміст однієї комірки пам'яті. За замовчуванням під час виконання команди *MOVSB* пересилається вміст комірки пам'яті з адресою *DS*:*SI* у комірку пам'яті з адресою *ES*:*DI*.

Приклад 2.33. Переслати слово, молодший байт якого зберігається у комірці пам'яті з адресою *DS*:*SI*, а старший байт — у комірці з адресою *DS*:*SI* + 1, у комірку пам'яті з адресою *ES*:*DI*.

За командою

MOVSW

(*MOVe Serial Word* — перемістити рядок слів) виконується пересилання слова (за замовчуванням використовуються ті самі адреси комірок, що й у прикладі 2.32).

Приклад 2.34. Переслати масив, що складається з 100 слів, з області пам'яті, яка має початкову адресу *DS*:*SI* (*DS* = 2000*H*, *SI* = 1000*H*) в область пам'яті, що має початкову адресу *ES*:*DI* (*ES* = 3000*H*, *DI* = 4000*H*).

За командою

REP MOVSW

(*REPeat MOVSW*) виконується пересилання слів, кількість яких зберігається за замовчуванням у регістрі *CX*.

Пересилання масиву здійснюється виконанням такої послідовності команд:

```
MOV   CX,100      ; Занести у CX значення довжини масиву
CLD               ; Скинути прапорець напряму для автоінкре-
                  ; ментації
                  ; адрес масивів
MOV   AX,2000H    ; Занести в регістри DS і SI адреси сегмента та
MOV   DS,AX       ; зміщення джерела
MOV   SI,1000H    ;
MOV   AX,3000H    ; Занести в регістри ES і DI адреси сегмента та
MOV   ES,AX       ; Зміщення регістра-приймача даних
MOV   DI,1000H    ;
REP   MOVSW       ; Переслати рядок слів
```

Розглянемо приклади виконання групи *команд передавання керування*. Команди передавання керування зазвичай змінюють уміст покажчика команд *IP*, а деякі ще й вміст регістра сегмента команд *CS*. За допомогою цих команд можна змінити послідовність виконання команд у програмі, оскільки регістр *CS* містить базову адресу поточного сегмента команд, з якого вибираються команди, а регістр *IP* — адресу, яка задає зміщення команди щодо початку сегмента команд. Після виконання команди передавання керування пристрій керування МП,

використовуючи новий вміст регістрів *CS* і *IP*, вибирає з пам'яті наступну команду, отже, виконується переоформлення черги команд, які надходять у блок операційного пристрою (див. рис. 2.11).

До цієї групи команд належать: команди безумовного та умовного переходів; команди викликів підпрограм та повернень з підпрограм; команди циклів; команди переривань.

Приклад 2.35. Виконати безумовний внутрішньосегментний перехід.

За командою

JMP NEAR LABEL

або

JMP LABEL

(*JuMP NEAR* — стрибок неподалік) здійснюється перехід до виконання команд, перша з яких позначена міткою *LABEL*, що знаходиться у поточному сегменті кодів, тобто близько 64 Кбайт. Якщо тип переходу не вказано, за замовчуванням виконується тип *NEAR*. Під час переходу в межах сегмента змінюється вміст програмного лічильника *IP* ← *'LABEL'*, де *LABEL* — символічна адреса або мітка, а *'LABEL'* — зміщення у сегменті кодів цієї мітки.

Приклад 2.36. Виконати безумовний міжсегментний перехід.

За командою

JMP FAR LABEL

(*JuMP FAR* — стрибок далеко) здійснюється перехід до виконання команд, першу з яких позначено міткою *LABEL*. Ця мітка знаходиться у межах всієї пам'яті ємністю 1 Мбайт. У разі міжсегментного переходу змінюється як вміст програмного лічильника *IP*, так і вміст сегментного регістра кодів *CS* згідно з міткою *LABEL*.

Приклад 2.37. Виконати умовний перехід залежно від стану прапорця *CF*.

За командою

JC LABEL

(*Jump if Carry* — стрибок, якщо є перенесення) здійснюється перехід виконання команд на мітку *LABEL* лише тоді, коли прапорець *CF* = 1, а якщо прапорець *CF* = 0, здійснюється перехід до виконання команди, наступної після команди умовного переходу.

Приклад 2.38. Викликати підпрограму, розташовану у пам'яті ємністю 1 Мбайт із символічною адресою *NAME*.

Для переходу до виконання підпрограми використовують команду *CALL*. Під час виконання команди *CALL* змінюються значення регістрів *CS* і *IP*. Перед виконанням команди запам'ятовуються значення цих регістрів, щоб після виконання підпрограми можна було повернутися до виконання основної програми. Запам'ятовування значень регістрів відбувається у стеку.

За командою

CALL FAR NAME

(*CALL* — виклик) викликається підпрограма з адресою *NAME:*

- вміст *SP* зменшується на 2;
- у комірку пам'яті з адресою *SS:SP* пересилається вміст регістра *CS*;
- вміст *SP* зменшується на 2;

• у комірку пам'яті з адресою *SS:SP* пересилається вміст регістра *IP*;

• в *IP* і *CS* завантажуються нові значення, які відповідають символічній адресі *NAME*.

У результаті цих дій у стеку запам'ятовується вміст регістрів *CS* і *IP*, тобто повна адреса *CS:IP* тієї команди, яку треба буде виконати після закінчення підпрограми *NAME*. Останньою командою підпрограми, що викликається, є команда *RET FAR*. За цією командою зі стеку витягуються два слова, які були записані під час виклику підпрограми, і заносяться у регістри *CS* і *IP*, тобто відновлюється поточна адреса *CS:IP* команди основної програми.

Приклад 2.39. Виконати послідовність команд 100 разів.

Для повторення виконання послідовності команд 100 разів треба в регістрі *CX* задати кількість повторень

MOV CX,100.

Потім записується послідовність команд, перша з яких позначається міткою:

*M*1: <послідовність команд>

та виконується команда

*LOOP M*1

(*LOOP* — петля). Команда *LOOP M*1 зменшує вміст *CX* на одиницю, а потім порівнює його з нулем. Якщо $CX \neq 0$,то здійснюється перехід до виконання команди з міткою *M*1, а якщо $CX = 0$, то виконується команда, наступна за *LOOP M*1 (вихід із циклу).

До команд циклів належать:

*LOOPE M*1

(*LOOP if Equal* — петля, якщо дорівнює) та

*LOOPNE M*1

(*LOOP if Not Equal* — петля, якщо не дорівнює). Ці команди реалізують вихід з циклу, якщо або $CX = 0$, або виконується додаткова умова: значення прапорця $ZF = 1$ або $ZF = 0$. Прапорець *ZF* може бути встановлений або відбутися його скидання в результаті виконання однієї з команд циклу.

Приклад 2.40. Перейти до підпрограми обробки переривання типу $n = 8$.

Мікропроцесор *i*8086 обробляє до 256 типів переривань. У спеціальній області пам'яті, яка розташована за початковою адресою 0000:0000 і називається *картою векторів переривань*, записано точки входів у підпрограми обробки переривань. За адресою 0000:4 × <номер переривання> зберігаються значення *IP*, за адресою 0000:4 × <номер переривання>+2 зберігаються значення *CS*. Нові значення *CS:IP* визначають адресу першої команди підпрограми оброблення переривання.

Команда

INT 8

ініціює таку послідовність дій:

• відбувається скидання прапорців *IF* і *TF*, що забороняє переривання та покомандну роботу МП;

• вміст *SP* зменшується на 2;
• у комірку пам'яті з адресою *SS*:*SP* пересилається вміст регістра прапорців *F*;
• вміст *SP* зменшується на 2;
• в комірку пам'яті з адресою *SS*:*SP* пересилається вміст регістра *CS*;
• вміст *SP* зменшується на 2;
• у комірку пам'яті з адресою *SS*:*SP* пересилається вміст регістра *IP*;
• номер переривання множиться на 4: $4 \times 8 = 20H$;
• у регістри *IP* і *CS* завантажуються нові значення з карти векторів переривань (початкова адреса $0000H$):

$$IP \leftarrow [0000{:}0020H],$$

$$CS \leftarrow [0000{:}0022H].$$

У результаті цих дій здійснюється міжсегментний непрямий виклик підпрограми обробки переривання, причому адреса підпрограми однозначно визначається номером переривання.

Отже, за командою *INT* 8 у стеку записується вміст регістрів *IP*, *CS* і *F*, а потім у регістри *IP* і *CS* записуються нові значення з карти векторів переривань; МП переходить до виконання підпрограми обробки переривання з номером 8.

Виконання команди *INT n* може бути ініційоване як програмно, так і апаратно. У першому випадку машинний код команди *INT* зчитується з програмної пам'яті, у другому — машинний код команди *INT* формує на шині даних систему переривань.

Розглянемо приклади виконання групи *команд керування станом МП*. До цієї групи належать такі команди: скидання, встановлення, інверсії прапорців, команда *ESC* (перемикання на співпроцесор), *LOCK* (захоплення шини), *NOP* (немає операції), *HLT* (зупинка), *WAIT* (очікування).

Приклад 2.41. Провести скидання прапорця напряму *DF* у стан логічного нуля.

За командою

```
CLD     ; DF ← 0
```

(*CLear DF* — очищення *DF*) відбувається скидання прапорця *DF* у стан логічного нуля.

Приклад 2.42. Установити прапорець напряму *DF* у стан логічної одиниці.

За командою

```
STD     ; DF ← 1
```

(*SeT DF* — встановлення *DF*) прапорець *DF* встановлюється у стан логічної одиниці.

2.4. Побудова модуля центрального процесора на базі *i*8086

Для побудови модуля ЦП слід забезпечити синхронізацію роботи системи та узгодження роботи ЦП із системною шиною.

Схема синхронізації. Для синхронізації використовується генератор тактових імпульсів *i*8284, який генерує сигнали

синхронізації для центрального процесора й периферійних пристроїв, а також синхронізує зовнішні сигнали готовності *READY* і початкового встановлення *RESET* з тактовими сигналами МП. Генератор тактових імпульсів (рис. 2.25) містить подільники частоти на 3 і на 2 та логіку керування сигналів скидання і готовності. Робота ЗГ стабілізується кварцовим резонатором, який під'єднується до входів *X*1, *X*2 (див. рис. 2.25). Вхід *TANK* використовується для додаткового під'єднання паралельного резонансного LC-контура, що дає змогу працювати на вищих гармоніках кварцового резонатора. При цьому опорна частота задавального генератора (ЗГ) визначається параметрами контура і дорівнює $\frac{1}{2\pi\sqrt{LC}}$. Вхід $F/\overline{C}$ дозволяє обрати зовнішній $(F/\overline{C}=1)$ або внутрішній $(F/\overline{C}=0)$ генератор. У разі обрання зовнішнього генератора з частотою імпульсів *FEFI* його підключають до входу *EFI*.

Схема формування тактових імпульсів формує такі сигнали: *CLK* — тактової частоти F_{CLK} для ЦП, P_{CLK} — тактової частоти F_{PCLK} для керування периферійними ВІС, *OSC* — тактової частоти ЗГ, необхідні для керування пристроями та контролю частоти. Частоти цих сигналів пов'язані певними співвідношеннями: $F_{OSC} = 3F_{CLK} = 6F_{PCLK}$ у режимі внутрішнього генератора та $F_{EFI} = 3F_{CLK} = 6F_{PCLK}$ у режимі зовнішнього генератора. Вихідний сигнал *CLK* формується одним з трьох способів: 1) з коливань основної час-

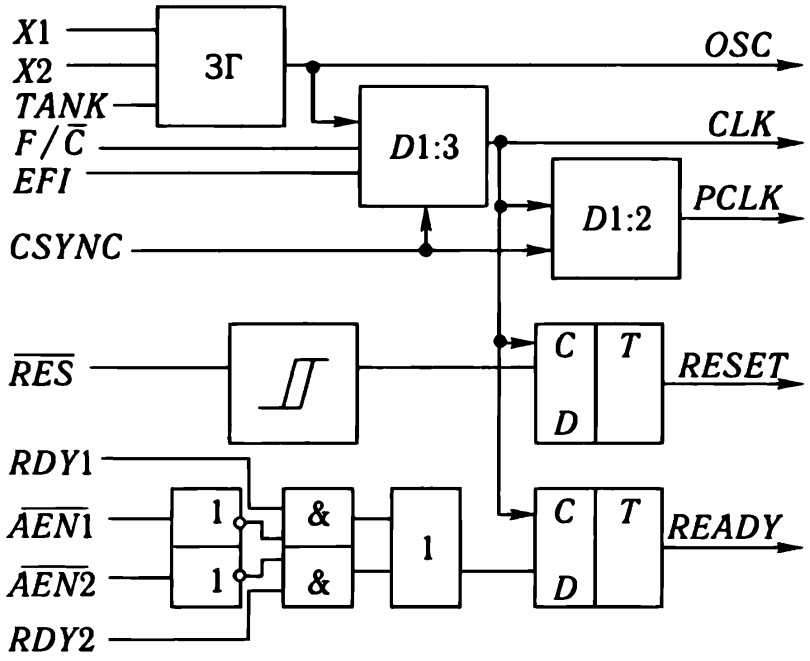


Рис. 2.25. Структурна схема ВІС генератора *i*8284

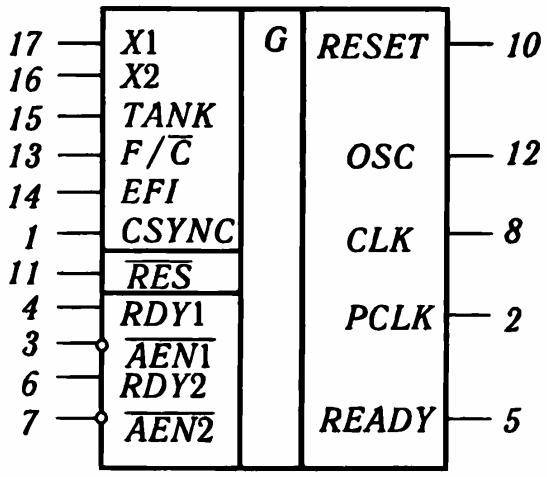


Рис. 2.26. Графічне позначення ВІС генератора *i*8284

тоти кварцового резонатора, під'єднаного до входів *X*1 і *X*2; 2) з третьої гармоніки кварцового резонатора, що виділяється LC-фільтром, з'єднаним із входом *TANK*; 3) від зовнішнього генератора, підключеного до входу *EFI*. Схема формування тактових імпульсів має вхід зовнішньої синхронізації *CSYNC*, за допомогою якого можна синхронізувати роботу кількох генераторів тактових імпульсів, що входять до складу системи. Сигнал *CSYNC* впливає також на подільник частоти на 2: якщо *CSYNC* = 0, робота подільника припиняється, а якщо *CSYNC* = 1, — відновлюється.

Вихідний сигнал *READY* використовують для підтвердження готовності до обміну. Високий рівень цього сигналу вказує на наявність даних на *DB*. Схему формування сигналу *READY* побудовано так, щоб спростити вмикання системи в інтерфейсну шину стандарта *Multibus*. Вона містить дві ідентичні пари сигналів *RDY1*, $\overline{AEN1}$ та *RDY2*, $\overline{AEN2}$, об'єднаних схемою АБО. Сигнали *RDY*1 та *RDY*2 формуються елементами, що входять до складу системи, і свідчать про їхню готовність до обміну. Сигнали $\overline{AEN1}$ та $\overline{AEN2}$ дають змогу формувати сигнал *READY* за сигналами *RDY*1 і *RDY*2, підтверджуючи адресацію елементів.

Схема формування вихідного сигналу скидання *RESET* має на вході тригер Шмітта, а на виході — тригер, що формує фронт сигналу *RESET* по зрізу сигналу *CLK*. Зазвичай до входу $\overline{RES}$ під'єднується RC-коло, що забезпечує автоматичне формування сигналу під час вмикання джерела живлення.

Графічне позначення генератора наведено на рис. 2.26.

Інтерфейс ЦП із системною шиною виконує такі функції:

- демультиплексування шини адреси-даних (розподіл її на шину адреси *AB* та шину даних *DB*);
- буферизацію шин (збільшення навантажувальної здатності ліній шин та забезпечення можливості їхнього переходу в *z*-стан);
- формування сигналів керування.

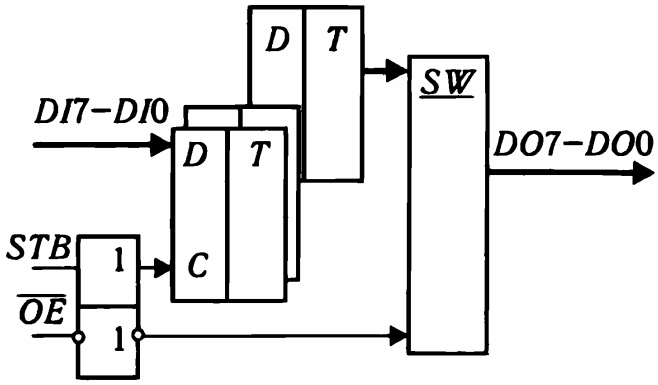


Рис. 2.27. Структурна схема буферного регістра

Виконання першої функції здійснюється за допомогою регістрів-фіксаторів, наприклад буферних регістрів *i*8282, *i*8283. Узагальнена структурна схема регістра-замка (рис. 2.27) містить вісім *D*-тригерів з вихідними схемами *SW*, які мають три стани. Сигнали запису інформації *STB* і дозволу вибірки $\overline{OE}$ є спільними для всіх тригерів ВІС. У буферному регістрі *i*8282 (рис. 2.28) до схем *SW* під'єднані прямі виходи *D*-тригерів, а в буферному регістрі *i*8283 (рис. 2.29) — інверсні виходи. Якщо сигнал має високий рівень, то на вході *STB* стан вхідних ліній *DI*7 — *DI*0 передається на вихідні лінії *DO*7 — *DO*0. Запис інформації у *D*-тригерах здійснюється по зрізу сигналу *STB*. Малий вхідний і досить великий вихідний струми дають змогу використовувати ВІС буферних регістрів як регістри-фіксатори або шинні формувачі. Під час використання буферних регістрів як шинних формувачів вхід *STB* з'єднується з виводом живлення +5 В через резистор з опором 1 кОм, а вхід $\overline{OE}$ — зі спільною шиною.

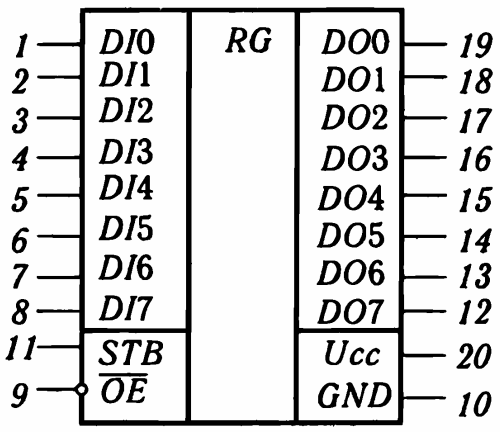


Рис. 2.28. Графічне позначення буферного регістра *i*8282

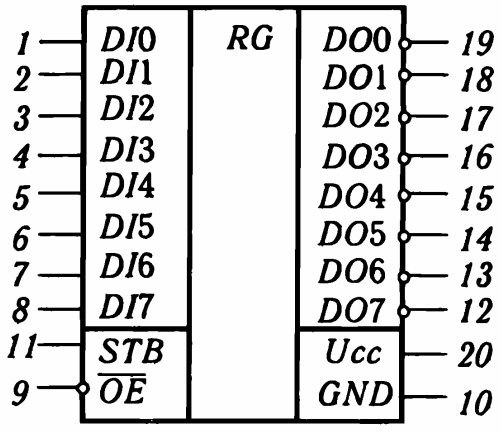


Рис. 2.29. Графічне позначення буферного регістра *i*8283

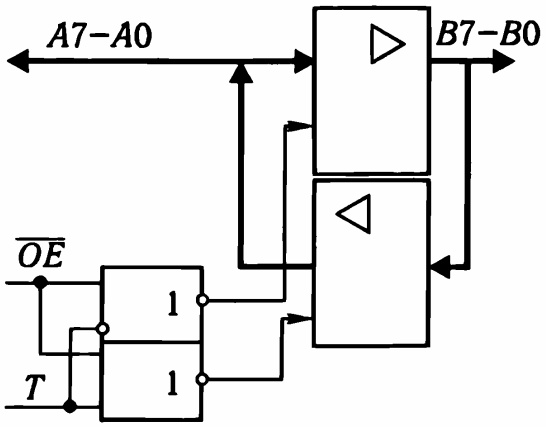


Рис. 2.30. Структурна схема шинного формувача

Для збільшення навантажувальної здатності двонапрямленої шини даних використовують 8-розрядні шинні формувачі *i*8286, *i*8287. Формувач *i*8286 дані не інвертує, а формувач *i*8287 їх інвертує.

Структурна схема шинного формувача (рис. 2.30) містить вісім однакових функціональних блоків з трьома станами і спільними сигналом керування напрямом передавання T та сигналом дозволу передавання $\overline{OE}$.

За низького рівня сигналу T ($T = 0$) здійснюється передавання даних з ліній $B7-B0$ на лінії $A7-A0$, а за високого рівня сигналу ($T = 1$) — передавання з ліній $A7-A0$ на лінії $B7-B0$. Якщо $\overline{OE} = 0$, передавання дозволяється, а якщо $\overline{OE} = 1$, — забороняється. Графічні позначення формувачів *i*8286, *i*8287 відповідно наведено на рис. 2.31 і 2.32.

На рис. 2.33 показано приклад функціональної схеми модуля ЦП для однопроцесорних систем). Мікропроцесор *i*8086 ввімкнений у мінімальному режимі. Схема синхронізації реалізована на базі ВІС тактового генератора *i*8284, на вхід *RDY*1 якої подається сигнал готовності зовнішніх пристроїв або пам'яті для обміну. У мінімальному режимі використовується одна шина, тому вхід *RDY*2 з'єднаний через резистор з виводом живлення. Демультиплексування шини адреси-даних і шини адреси-стану на дві шини здійснюється за допомогою трьох буферних регістрів *i*8282.

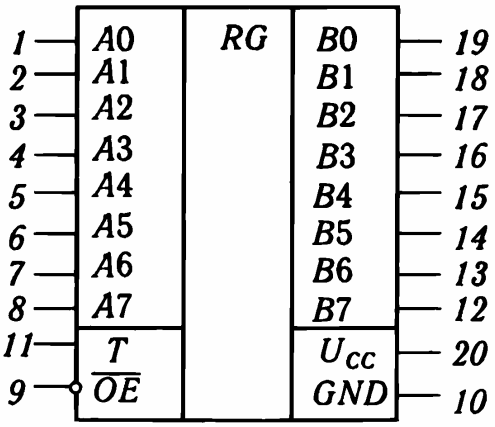


Рис. 2.31. Графічне позначення шинного формувача *i*8286

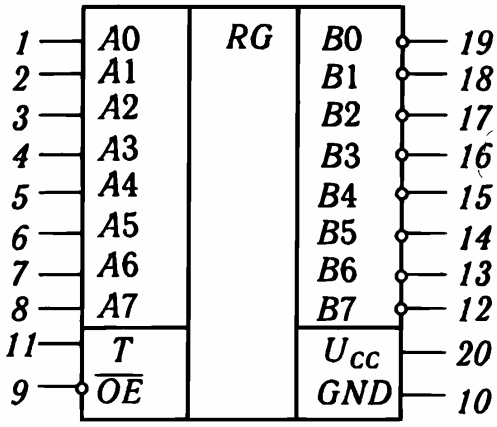


Рис. 2.32. Графічне позначення шинного формувача *i*8287

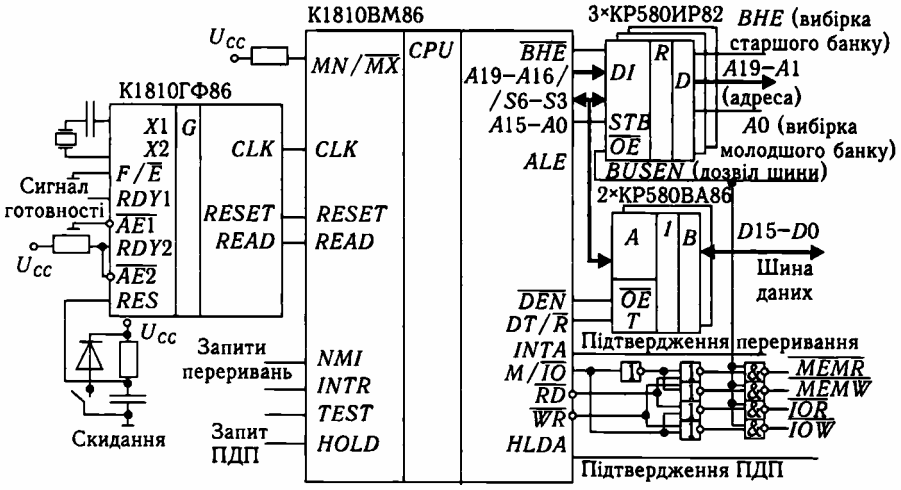


Рис. 2.33. Функціональна схема модуля центрального процессора

Зазначимо, що сигнал дозволу старшого байта $\overline{BHE}$, що з'являється водночас із дійсною адресою, також фіксується в одному з розрядів регістрів-фіксаторів. Сигнали $\overline{BHE}$ і $A0$ використовуються для вибірки банків системи пам'яті. Формувач 16-розрядної шини даних виконано на двох ВІС шинних формувачів *i*8286.

У мінімальному режимі процесор формує керуючі сигнали шин формувачів і регістрів-фіксаторів, а також сигнали $M/\overline{IO}$, $\overline{RD}$, $\overline{WR}$, з яких за допомогою логічних елементів формуються чотири сигнали керування читанням-записом для пам'яті і ПВВ. Шини адреси, даних і керування переводяться у *z*-стан сигналом $\overline{BUSEN}$, що формує контролер прямого доступу до пам'яті.

Контрольні запитання

1. Розкажіть про призначення регістра команд, акумулятора, блока РЗП.
2. Яке призначення прапорців у МП?
3. Які керуючі сигнали використовує МП для забезпечення роботи з пристроями, швидкодія яких значно менша, ніж швидкодія самого МП?
4. Які керуючі сигнали використовує МП для забезпечення роботи з пристроями, швидкодія яких перевищує швидкодію МП?
5. Які дії виконує процесор за сигналом *RESET*?
6. Які режими адресації використовуються у 8-розрядних МП?
7. Яку інформацію несе байт стану?
8. Які дії відбуваються в тактах *T*1 і *T*2 будь-якого машинного циклу?
9. Назвіть можливі типи машинних циклів.
10. Назвіть послідовність дій процесора у машинному циклі ПІДТВЕРДЖЕННЯ ПЕРЕРИВАННЯ.
11. Які дії виконує МП після надходження запиту переривання у режимі переривання?
12. Які дії виконує МП після вмикання напруги живлення? Назвіть можливі способи виходу з режиму зупинки.
13. Які характерні особливості мінімального і максимального режимів роботи МП *i*8086?
14. Назвіть існуючі формати даних МП *i*8086.
15. Наведіть приклади упакованого і розпакованого двійково-десяткових чисел.
16. Як у МП подати від'ємні числа?
17. Поясніть принцип конвеєрної архітектури.
18. Назвіть функції операційного пристрою і шинного інтерфейсу.
19. Як визначити тип, початок і кінець циклу шини за допомогою ліній стану?
20. Які блоки МП беруть участь у формуванні 20-розрядної фізичної адреси?

21. Обчисліть 20-розрядну фізичну адресу *DS*:*SI*, якщо *DS* = 1234*H*, *SI* = 5678*H*.
22. Підберіть дві пари 16-розрядних логічних адрес, які є еквівалентними фізичній адресі 12008*H*.
23. Які групи регістрів входять до програмної моделі МП?
24. Які сегментні регістри за замовчуванням адресують початок сегментів кодів, стеку, даних?
25. Яке призначення регістра прапорців?
26. Наведіть приклад виконання команди, після якої встановлюється прапорець знака.
27. Наведіть приклад виконання команди, після якої встановлюється прапорець паритету.
28. Наведіть приклад виконання команди, після якої встановлюється прапорець нульового результату.
29. Наведіть приклади команд введення-виведення з прямою адресацією 8- і 16-розрядного портів.
30. Які є типи адресації операндів у пам'яті?
31. Як обчислити ефективну адресу операнда для різних типів адресації?
32. Назвіть та схарактеризуйте існуючі типи циклів шини.
33. Дайте визначення вектора переривань і карти векторів переривань.
34. Які дії виконує МП під час переходу на підпрограму оброблення переривань?
35. Назвіть та схарактеризуйте існуючі типи переривань МП *i*8086.
36. На які групи поділяють команди МП *i*8086?
37. Які групи команд не впливають на встановлення прапорців?
38. Розкажіть про значення прапорців, які встановлюються під час додавання чисел 25*H* і 97*H*?
39. Які дії виконує МП після виклику процедури типу *FAR* та типу *NEAR*?
40. Які прапорці змінюються за викликів переривань?
41. Поясніть призначення вхідних і вихідних сигналів схеми синхронізації.
42. Опишіть функції ВІС генератора *i*8284.
43. Опишіть функції інтерфейсу центрального процесора з системною шиною.
44. Поясніть принцип функціонування схеми буферного регістра.
45. Поясніть принцип функціонування шинного формувача.

Розділ 3 ОДНОКРИСТАЛЬНІ УНІВЕРСАЛЬНІ МІКРОПРОЦЕСОРИ (СТАРШІ МОДЕЛІ)

3.1. Мікропроцесор *i*80286

Загальні відомості. Мікропроцесор *i*80286 належить до другого покоління 16-розрядних МП. Він виконаний за технологією 1,5 мкм, містить 134 000 транзисторів і працює з тактовою частотою 12,5 МГц. Після вдосконалення архітектури швидкодія МП *i*80286 у 6 разів вища, ніж МП *i*8086 з тактовою частотою 5 МГц. Розрядність регістрів дорівнює 16. Шина адреси 24-розрядна, що дає змогу адресувати 224 = = 16 Мбайт пам'яті. Простір адрес введення-виведення становить 64 Кбайт. Система команд містить усі команди *i*8086, кілька нових команд загального призначення та команди керування захистом. Мікропроцесор *i*80280 має спеціальні засоби для роботи у системах з багатьма користувачами та багатозадачних режимах. Його найістотнішою відмінністю від мікропроцесорів серії *i*8086/88 є механізм керування адресацією пам'яті, що забезпечує чотирирівневу систему захисту та підтримку віртуальної пам'яті. Спеціальні засоби призначено для підтримки механізму перемикання задач. Мікропроцесор *i*80286 має засоби контролю за переходом через межу сегмента, які працюють у реальному режимі.

Мікропроцесор може працювати у двох режимах:

• *8080 Real Address Mode,* або *Real Mode,* — режим реальної адресації, або реальний режим. У цьому режимі МП *i*80286 фактично є високошвидкісним МП *i*8086 і адресує 1 Мбайт пам'яті;

• *Protected Virtual Address Mode,* або *Protected Mode,* — захищений режим віртуальної адресації або просто захищений режим. У цьому режимі МП адресує до 16 Мбайт пам'яті, а за використання механізму сторінкової адресації до 1 Гбайт віртуальної пам'яті кожної задачі.

Перемикання у захищений режим здійснюється швидко — однією командою (із заздалегідь підготовленими таблицями дескрипторів), а в режим реальної адресації — повільно, лише

через апаратне скидання процесора. У MS DOS використовується реальний режим. Захищений режим використовується в операційних системах на зразок XENIX, UNIX, OS/2, NetWare286, MS Windows.

Для процесора *i*80286 можливі 256 різних типів переривань. Відрізняється від системи переривань МП *i*8086 перериванням під час виникнення особливих умов під час виконання команд, наприклад за розміщення двобайтового операнда в останній комірці сегмента даних зі зміщенням *FFFFH*. Таке переривання називають *особливим випадком,* або *винятком.* На відміну від переривань після оброблення винятків (крім винятку 9, що стосується співпроцесора) керування передається знову тій самій команді (включаючи всі префікси), що зумовила переривання. Після усунення умов, що спричинили виняток, відбувається повторне виконання команди.

Організація пам'яті. У реальному режимі адресація пам'яті переважно така сама, як і в МП *i*8086. Відмінність полягає у можливості використання додаткового блока пам'яті ємністю 64 Кбайт. Якщо у процесі виконання команди під час обчислення адреси комірки пам'яті виникає переповнення у двадцятий розряд шини адреси *A*20, процесор починає працювати з комірками пам'яті, адреси яких знаходяться в діапазоні 100 000*H* – 10*FFFFH*.

Приклад 3.1. Знайти значення фізичної адреси операнда в реальному режимі під час виконання команди *MOV AL*, [*SI*]; пересилання у регістр *AL* вмісту комірки пам'яті з адресою *DS*:*SI*, якщо вмістом сегментного регістра *DS* є число 0*F*802*H*, а вмістом регістра *SI* – 0*B*175*H*.

Для обчислення фізичної адреси до значення *DS* додамо чотири нулі праворуч:

$$DS(0000) = 1111\ 1000\ 0000\ 0010\ 0000B = 0F8020H.$$

Виконавши операцію додавання отриманої величини з вмістом регістра *SI*, дістанемо фізичну адресу:

```
  11111          11          — рядок перенесень
   1111 1000 0000 0010 0000
+
        1011 0001 0111 0101
 1 0000 0011 0001 1001 0101 = 103195H.
```

Отже, під час обчислення фізичної адреси отримане одиничне значення розряду *A*20 означає, що операнд розташується у другому мегабайті фізичної пам'яті.

Ще однією особливістю реального режиму *i*80286 є можливість контролю за переходом за межі сегмента. Під час

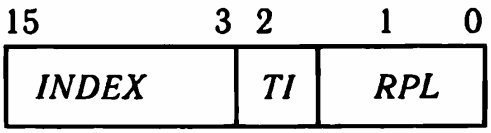


Рис. 3.1. Формат селектора сегмента

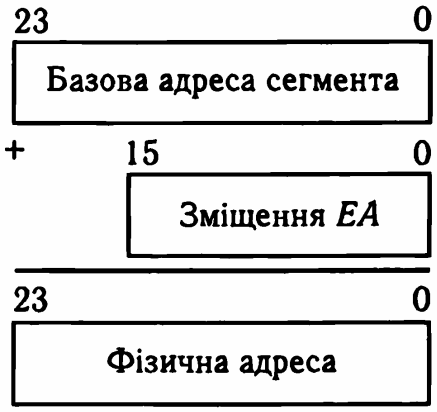


Рис. 3.2. Формування фізичної 24-розрядної адреси у захищеному режимі

адресації слова зі зміщенням 0*FFFFH* генерується виняток 13 (*Segment Overrun Exception*). За спроби виконання команди *ESC* з операндом пам'яті, що не вміщається у сегменті, генерується виняток 9 — *Processor Extension Segment Overrun Interrupt*.

У захищеному режимі також використовується сегментна адресація; кількість сегментів може бути від 1 до 16 Мбайт, довжина сегментів задається і може варіювати від 1 до 64 Кбайт, задаються атрибути або права доступу до сегмента (дозвіл запису або лише читання, рівні привілеїв тощо). Кожний сегмент характеризується 8-розрядною структурою даних — *дескриптором сегмента,* що містить інформацію про базову адресу сегмента, його межу та атрибути. Дескриптори розміщені у спеціальних таблицях — глобальній дескрипторній таблиці *GDT* або локальній дескрипторній таблиці *LDT*, які зберігаються в ОЗП. Незалежно від рівня привілею програма не може звертатися до сегмента доти, доки він не описаний у дескрипторній таблиці.

У захищеному режимі вміст сегментних регістрів називають *селекторами сегментів.* Їх використовують для пошуку базової (початкової) адреси сегмента в одній з дескрипторних таблиць. Формат селектора показано на рис. 3.1.

Селектори, що завантажуються у 16-розрядні сегментні регістри *CS*, *DS*, *SS*, *ES*, мають три поля:

- *RPL* (*Requested Privilege Level*) (біти 0 і 1) — запрошений рівень привілеїв сегмента;
- *TI* (*Table Indicator*) (біт 2) — індикатор використання таблиці дескрипторів (якщо *TI* = 1, використовується глобальна таблиця, а якщо *TI* = 0, — локальна);
- *INDEX* (біти 3—15) — номер дескриптора у таблиці.

Глобальна дескрипторна таблиця єдина. Вона містить дескриптори для всіх задач, які виконує МП у багатозадачному режимі. Локальних дескрипторних таблиць може бути кілька, причому для кожної задачі можна задати свою локальну дескрипторну таблицю.

Фізичну 24-розрядну адресу операнда знаходять додаванням початкової 24-розрядної адреси сегмента з дескрипторної таблиці та адреси-зміщення, яка вказується у команді. Формування фізичної 24-розрядної адреси у захищеному режимі наведено на рис. 3.2.

Приклад 3.2. Знайти значення фізичної адреси комірки пам'яті [*DS*:*SI*], якщо базова адреса сегмента даних дорівнює 456 789*H*, а вміст регістра *SI* — 1234*H*.

Виконавши операцію додавання 24-розрядної базової адреси з адресою-зміщенням, тобто з вмістом регістра *SI*, отримаємо фізичну адресу:

```
    0100 0101 0110 0111 1000 1001
+               0001 0010 0011 0100
    0100 0101 0111 1001 1011 1101  = 4579BBH.
```

Отже, фізична адреса дорівнює 4579*BBH*.

Програмна модель. До програмної моделі МП *i*80286 (рис. 3.3) входять 19 програмно-доступних регістрів і 6 недоступних (тіньових). Із 19 програмно-доступних регістрів МП *i*80286 14 повторюють регістри процесора *i*8086 (див. п. 3.5). П'ять нових регістрів такі:

40-розрядний регістр *GDTR* (*Global Descriptor Table Register*) — регістр глобальної дескрипторної таблиці. Призначений для задання розміщення глобальної дескрипторної таблиці в пам'яті. Регістр *GDTR* безпосередньо містить базову адресу і значення межі таблиці, яке задає її розмір. Розмір таблиці більший, ніж значення межі, на одиницю;

16-розрядний регістр *LDTR* (*Local Descriptor Table Register*) — регістр-селектор локальної дескрипторної таблиці. Його вміст вказує, де в глобальній дескрипторній таблиці знаходиться інформація про початкову адресу, межу і права доступу до локальної таблиці. Ця інформація переписується у тіньовий програмно-недоступний регістр;

40-розрядний регістр *IDTR* (*Interrupt Descriptor Table Register*) — регістр дескрипторної таблиці переривань. Завдяки цьому регістру в МП *i*80286 з'явилася можливість розміщувати таблицю векторів переривань у довільному місці ОЗП, а не з нульової адреси, як у МП *i*8086. Регістр *IDTR* за структурою аналогічний регістру *GDTR*;

16-розрядний регістр задачі *TR* (*Task Register*) — регістр-селектор сегмента стану поточної задачі *TSS* (*Task State Segment*). Такі сегменти асоціюються з кожною задачею. Їхнє призначення — збереження контексту задачі під час перемикання задач;

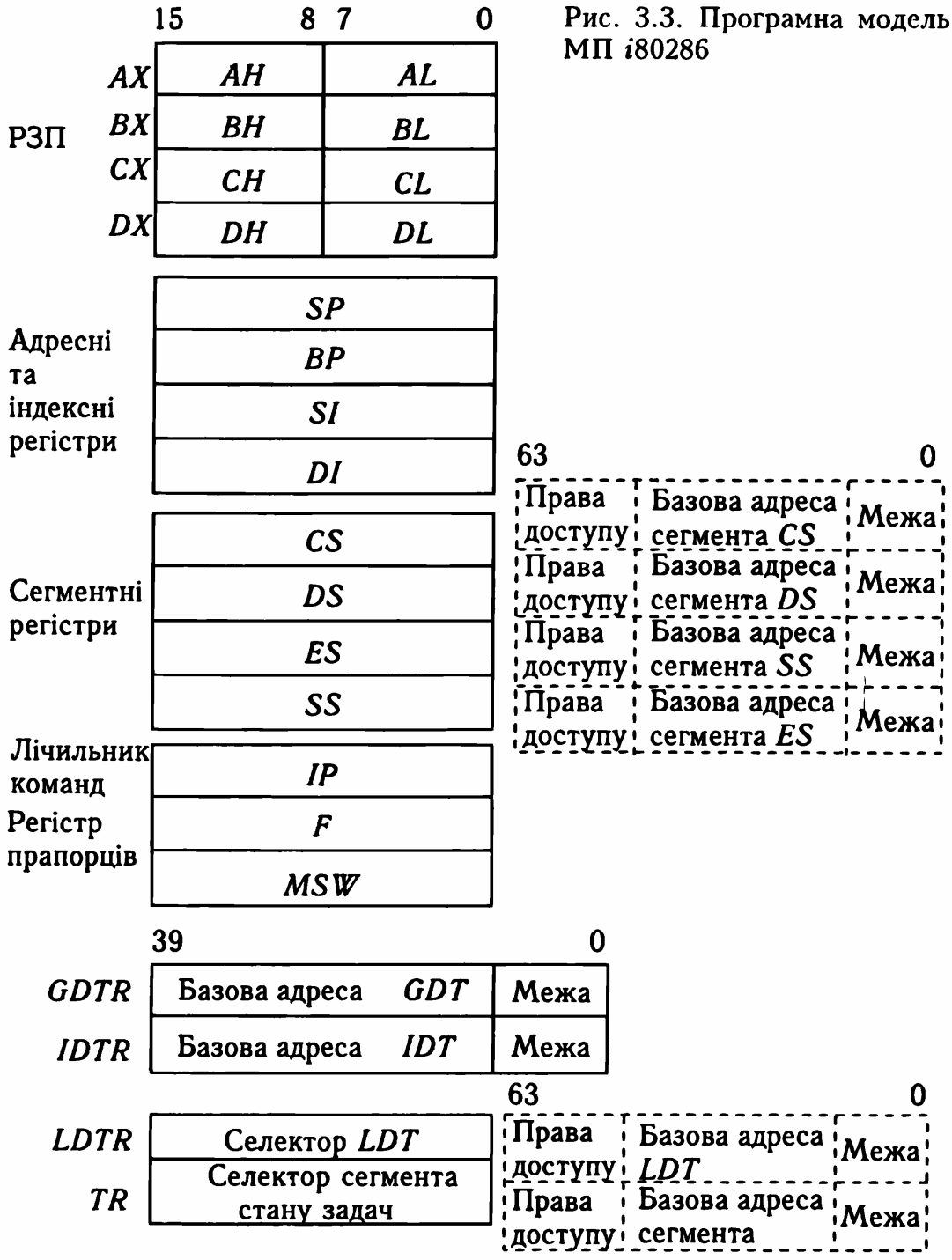


Рис. 3.3. Програмна модель МП *i*80286

16-розрядний регістр стану машини *MSW* (*Machine State Word*), що керує режимом процесора і містить такі біти (рис. 3.4):

PE (*Protecrion Enable*) — дозвіл захисту. Встановлення цього прапорця переводить процесор у захищений режим. Повернення до реального режиму можливе лише за сигналом *RESET*.

MP (*Monitor Processor Extension*) — наявність співпроцесора. Якщо арифметичний співпроцесор *i*80287 з'єднаний

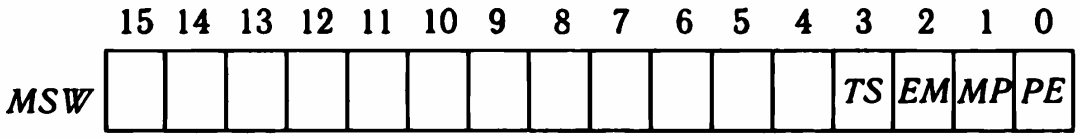


Рис. 3.4. Регістр стану машини *MSW*

з процесором *i*80286, то за ініціалізації операційна система має встановити цей біт у стан логічної одиниці. Тоді під час виконання команд *WAIT* та *ESC* і значення *TS* = 1 мікропроцесор генеруватиме виняток 7.

EM (*Processor Extension Emulated*) — емуляція співпроцесора. Встановлення цього прапорця викликає виняток 7 після виконання команд арифметичного співпроцесора.

TS (*Task Switch*) — перемикання задач. Зі встановленням прапорця наступна команда, яка належить до співпроцесора, викличе виняток 7.

Тіньові регістри відіграють роль надоперативної пам'яті. Їхнє призначення — підвищення швидкодії роботи МП (на рис. 3.3 тіньові регістри показано штриховою лінією).

Крім того, у програмній моделі, на відміну від МП *i*8086, додано нові біти у регістрі прапорців і змінено використання сегментних регістрів у захищеному режимі. У регістрі прапорців біт 14 визначений як *NT* (*Nested Task Flag* — прапорець певної задачі), а біти 13—12 — як *IOPL* (*Input/Output Privilege Level* — двобітове поле рівня привілеїв введення-виведення). Ці прапорці діють лише в захищеному режимі. Прапорець *NT* встановлюється у стан логічної одиниці під час перемикання задач за допомогою команди *CALL*. Після виконання команди *IRET* перевіряється стан прапорця *NT* і якщо *NT* = 1, здійснюється перемикання, якщо ні, то виконується звичайне повернення з переривання. Поле *IOPL* вказує рівень привілею поточної задачі, за якого дозволяється виконання певних операцій.

Сегментні регістри CS, *SS*, *ES* і *DS* визначають початкові адреси сегментів. У реальному режимі 20-розрядна початкова адреса сегмента визначається як вміст 16-розрядного сегментного регістра, доповненого праворуч чотирма нульовими бітами. У захищеному режимі початкова 24-розрядна базова адреса сегмента знаходиться у дескрипторній таблиці в ОЗП, а вміст сегментних регістрів є селекторами, які вказують на тип таблиці та номер запису в ній (див. рис. 3.1).

Під час завантаження нового значення селектора дескриптори зчитуються з ОЗП і запам'ятовуються у внутрішніх програмно-недоступних або тіньових регістрах процесора. Це

дає змогу підвищити швидкодію процесора, оскільки значення базових адрес сегментів змінюються порівняно рідко.

Регістри *GDTR, LDTR, IDTR* задають розташування дескрипторних таблиць у пам'яті (рис. 3.5). На рис. 3.5 показано, що глобальна дескрипторна таблиця містить $N + 1$ дескрипторів, локальна — $K + 1$, дескрипторна таблиця переривань — $M + 1$. Початкова адреса глобальної дескрипторної таблиці визначається бітами 39—16 регістра *GDTR* та обчислюється додаванням значень початкової адреси і межі таблиці (біти 15—0). Аналогічно початкова і кінцева адреси таблиці переривань визначаються вмістом регістра *IDTR*. Для адресації дескрипторів локальної таблиці використовують вміст регістра *LDTR*, який є селектором. Він вказує на номер дескриптора у глобальній дескрипторній таблиці. Цей дескриптор завантажується у тіньовий регістр (на рис. 3.5 зображено штриховою лінією). Перша та остання адреси локальної дескрипторної таблиці визначаються вмістом цього регістра.

Приклад 3.3. Знайти значення, яке треба завантажити у регістр *GDTR*, щоб задати у пам'яті глобальну дескрипторну таблицю з 21 дескриптора і з початковою адресою 000100*H*.

Біти 39—16 регістра *GDTR* задають початкову 24-розрядну адресу, отже, мають дорівнювати 000100*H*. Кожний запис у таблиці займає

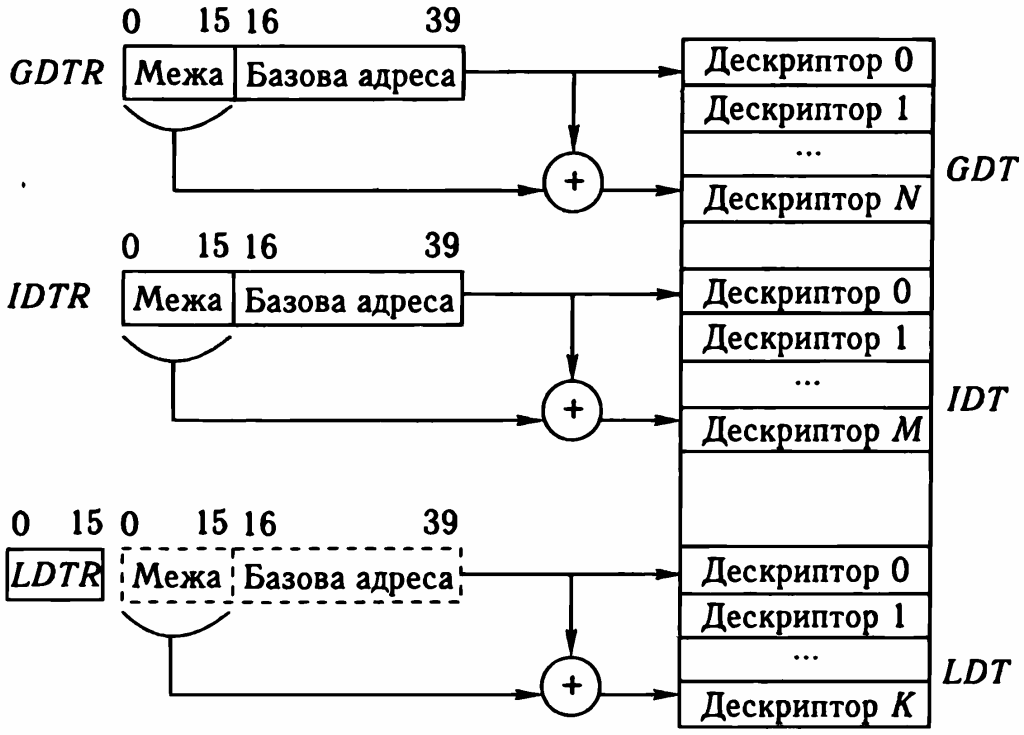


Рис. 3.5. Розміщення дескрипторних таблиць у пам'яті

8 байт, тому, адреса 21 дескриптора визначається як 000100*H* + + 20 · 8 = 0001A0*H*. У бітах 15-0 регістра *GDTR* треба розмістити число 0*A*0*H*, після чого вміст регістра *GDTR* становить

0001000*AH*.

Зазначимо, що команди завантаження регістрів таблиць *GDTR*, *LDTR*, *IDTR*, *LGDT*, *LLDT*, *LIDT* є привілейованими і виконуються у програмах з вищим рівнем пріоритету.

Адресний простір портів введення-виведення. Адресний простір портів МП *i*80286 такий самий, як і для МП *i*8086, тобто становить $64 \cdot 2^{10}$ одно- або $32 \cdot 2^{10}$ двобайтових портів. Додаткові рядкові команди *REP INSB/INSW*, *REP OUTSB/OUTSW* (табл. 3.1) забезпечують блокове оброблення зі швидкістю, що перевищує аналогічні операції в режимі ПДП. У захищеному режимі команди є привілейованими, тобто вони можуть виконуватися лише з певним рівнем привілею, що визначається полем *IOPL* регістра прапорців. Інакше викликається виняток 13 — порушення захисту.

Система команд. До системи команд МП *i*80286 входять усі команди МП *i*8086 і ряд додаткових (див. табл. 3.1). У табл. 3.1 використано ті самі позначення, що й у табл. 2.11.

Таблиця 3.1. **Додаткові команди МП *i*80286**

Мнемокод	Опис
	Команди роботи зі стеком
PUSH immed	Переміщення безпосередніх даних *immed* у стек
PUSHA	Переміщення у стек вмісту регістрів *AX*, *BX*, *CX*, *DX*, *SI*, *DI*, *BP*, *SP*
POPA	Переміщення даних зі стеку в регістри *AX*, *BX*, *CX*, *DX*, *SI*, *DI*, *BP*, *SP*
	Арифметичні команди
IMUL reg16, r/m	Множення вмісту *reg16* на вміст *r/m* (16 біт)
IMUL reg16, r/m, immed	Множення вмісту *r/m* на 16-бітовий безпосередній операнд і переміщення результату в *reg16*
	Рядкові команди
[*REP*] *INSB* ([*REP*] *INSW*)	Введення байта (слова) у комірку пам'яті *ES*: [*DI*] із порту з адресою *DX* з автоінкрементуванням (*DF* = 0) або автодекрементуванням (*DF* = 1) адреси. У разі використання префікса *REP* операція повторюється *CX* разів

Продовж. табл. 3.1

Мнемокод	Опис
[*REP*] *OUTSB* ([*REP*] *OUTSW*)	Виведення байта (слова) з комірки пам'яті *DS*: [*SI*] у порт з адресою *DX* з автоінкрементуванням (*DF* = 0) або автодекрементуванням (*DF* = 1) адреси. У разі використання префікса *REP* операція повторюється *CX* разів
	Команди переривань
*BOUND reg*16, *lmts*	Перевірка меж масиву — якщо знакове число у *reg*16 не знаходиться в заданих межах, виконується *INT* 5. Межі задаються у двох суміжних словах пам'яті за адресою *lmts*.
	Команди підтримки процедур
ENTER frmsiz, *frms*	Підготовка блока параметрів процедур (*frmsiz* — кількість байт для змінних процедури, *frms* — рівень укладення процедур)
LEAVE	Відміна чинності *ENTER* (відновлює значення вмісту регістрів *SP* і *BP*)
	Команди керування станом МП
CLTS	Скидання прапорця перемикання задач
SEG	Префікс заміни сегмента
	Команди керування захистом
LGDT src	Завантаження регістра *GDTR* з пам'яті (6 байт)*
SGDT dest	Збереження вмісту регістра *GDTR* у пам'яті (6 байт)*
LIDT src	Завантаження регістра *IDTR* з пам'яті (6 байт)*
SIDT dest	Збереження вмісту регістра *IDTR* у пам'яті (6 байт)*
LLDT src	Завантаження регістра *LDTR* з регістра або пам'яті *reg*/*mem*16
SLDT dest	Збереження вмісту регістра *LDTR* у *reg*/*mem*16
LMSW src	Завантаження регістра *MSW* з регістра або пам'яті *reg*/*mem*16
SMSW dest	Збереження регістра *MSW* у *reg*/*mem*16
LTR src	Завантаження регістра задачі з *reg*/*mem*16
STR dest	Збереження вмісту регістра задачі в *reg*/*mem*16
LAR dest, *src*	Завантаження старшого байта *dest* байтом прав доступу дескриптора *src*

Продовж. табл. 3.1

Мнемокод	Опис
LSL dest, src	Завантаження *dest* межею сегмента, дескриптор якого заданий *src*
*ARPL reg/mem*16, *reg*16	Вирівнювання *RPL* у селекторі до найбільшого числа з поточного рівня і заданого операндом
VERR seg	Верифікація читання: встановлення прапорця *ZF* у стан логічної одиниці (*ZF* = 1), для дозволу читання у сегменті *seg*
VERW seg	Верифікація запису: встановлення *ZF* = 1 для дозволу запису в сегмент *seg*

*Команди *LGDT*, *LIDT* завантажують у 40-розрядні регістри *GDTR*, *IDTR* п'ять байтів, але під час виконання команди передаються шість байтів: перші п'ять з них пересилаються у регістри, а шостий (з найбільшою адресою) ігнорується. Однак для сумісності з майбутніми розробками МП він повинен мати нульове значення. Команди *SGDT*, *SIDT* передають вміст регістрів *GDTR*, *IDTR* у пам'ять, при цьому передаються шість байтів: у п'яти знаходиться вміст регістра, а шостий — невизначений.

Приклад 3.4. Написати програму переведення МП у захищений режим адресації.

Для переведення МП у захищений режим адресації треба встановити біт *PE* — молодший біт регістра *MSW*. Однак перед установленням цього біта треба задати початкові значення регістрів таблиць *GDTR*, *LDTR*, *IDTR* значення елементів дескрипторних таблиць в ОЗП. Зазвичай таблиці переписують із ПЗП в ОЗП. Зазначимо, що таблиці мають знаходитися в ОЗП, оскільки під час виконання програми їхній вміст може модифікуватися.

Програма переведення МП має вигляд:

```
SMSW    BX          ; Збереження MSW у BX
OR      BX,0001H    ; Установлення молодшого біта у стан
                    ; логічної одиниці
LMSW    BX          ; Завантаження MSW
JMP     M1          ; Скидання черги команд
M1:                 ; Початок роботи у захищеному режимі
```

Короткий перехід *JMP M1* на наступну команду потрібний для скидання черги команд. У МП *i*80286 існує випереджальна вибірка команд. Якщо МП виконав команду *LMSW*, яка перевела його у захищений режим, а в черзі команд залишилися команди, які мали виконуватися у реальному режимі, то після переведення процесора у захищений режим команди реального режиму будуть декодовані неправильно.

Цикли шини. Мікропроцесор *i*80286 має шинний інтерфейс з 6-байтовою чергою команд, що забезпечує конвеєрну

адресацію (*Pipelined Addressing*). Це забезпечує вибірку кодів команд або даних із пам'яті з випередженням, що дає змогу почати фазу ідентифікації або адресації нового циклу, не дочекавшись закінчення попереднього.

Розрізняють шість типів циклів шини:

- ЧИТАННЯ ПАМ'ЯТІ;
- ЗАПИС У ПАМ'ЯТЬ;
- ЧИТАННЯ ПОРТУ;
- ЗАПИС У ПОРТ;
- ПІДТВЕРДЖЕННЯ ПЕРЕРИВАННЯ;
- СТОП-ЗУПИНКА.

Зазначимо, що слово з непарною адресою передається за два цикли шини, а з парною — за один. На рис. 3.6 показано цикли шини ЧИТАННЯ і ЗАПИС слова у комірку пам'яті з парною адресою. Зчитування слова з ОЗП здійснюється щонайменше за чотири періоди тактових імпульсів *CLK* або за два стани процесора (без урахування сигналу готовності *READY*). Кожний стан МП триває два такти *CLK*. Під час першого стану, позначеного через T_S, процесор виставляє на шину адреси значення адреси комірки пам'яті, з якої буде зчитуватися слово. Завдяки конвеєрній адресації адреса комірки виставляється з деяким випередженням, однак вона не зберігається на шині адреси впродовж циклу. Для сумісності

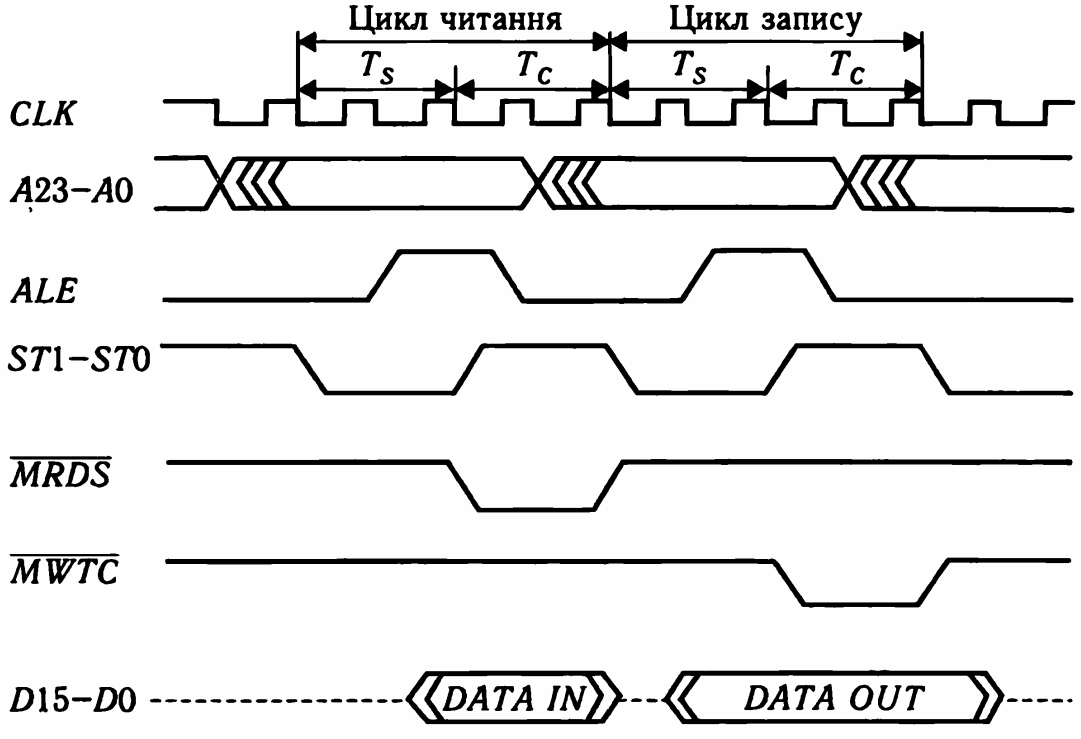


Рис. 3.6. Цикли шини ЧИТАННЯ і ЗАПИС слова у комірку пам'яті з парною адресою

з шиною *ISA* сигнали шини адреси запам'ятовуються у регістрах-фіксаторах за стробом *ALE*. Керуючі сигнали читання пам'яті $\overline{MRDS}$ і адресний строб *ALE* формуються системним контролером 82288 на початку другого стану T_C. Сигнали керування та адреси обробляються схемою керування пам'яттю, внаслідок чого, починаючи з середини другого стану T_C, на шині даних з'являється слово з ОЗП, і процесор зчитує його із шини даних. Тривалість циклу ЧИТАННЯ з парною адресою становить тривалість двох станів.

Під час зчитування слова з непарною адресою в першому циклі переноситься байт за старшою половиною шини даних $D15-D8$, а в другому — передається другий байт за молодшою половиною $D7-D0$. Цикл шини ЗАПИС У ПАМ'ЯТЬ з парною адресою також дорівнює тривалості двох станів. У першій половині циклу T_S виставляється адреса і дані, в другій — відбувається запис в ОЗП.

Виконання циклів шини ЧИТАННЯ ПОРТУ та ЗАПИС У ПОРТ аналогічне розглянутим вище циклам ЧИТАННЯ ПАМ'ЯТІ та ЗАПИС У ПАМ'ЯТЬ, однак у цьому випадку на шину $A15-A0$ виставляється адреса порту і замість сигналів читання або запису пам'яті генеруються сигнали $\overline{IOR}$ — для читання портів введення-виведення, або $\overline{IOW}$ — для запису в порти введення-виведення.

Цикл шини ПІДТВЕРДЖЕННЯ ПЕРЕРИВАННЯ (рис. 3.7) виконується у разі виникнення апаратного переривання (наявності активного рівня на виводі *INTR*). Цикл складається з двох процесорних циклів, поділених трьома тактами очікування T_w.

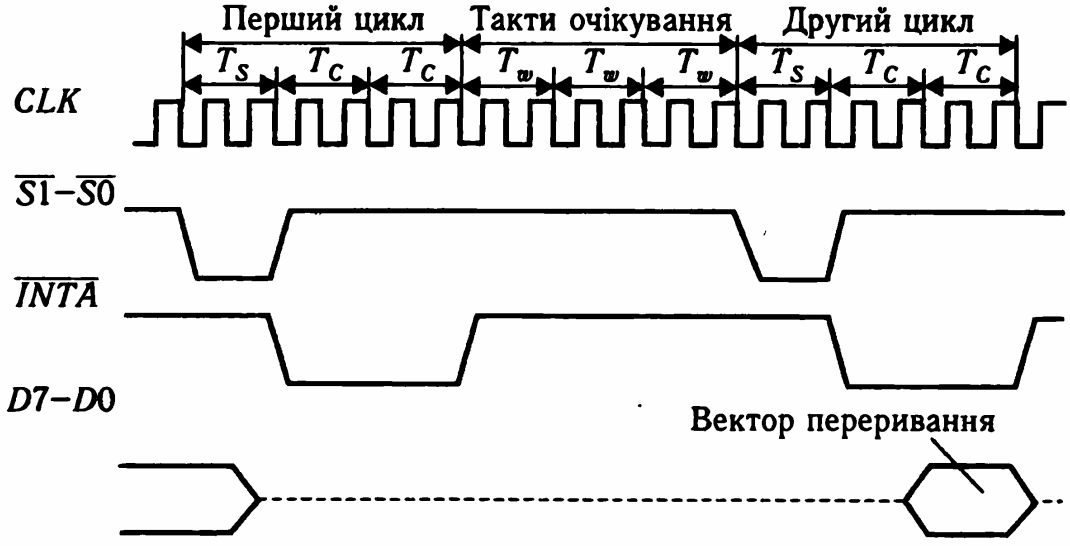


Рис. 3.7. Цикл шини ПІДТВЕРДЖЕННЯ ПЕРЕРИВАННЯ

Кожний процесорний цикл складається з трьох станів — T_S, T_C, T_C. Це потрібно для того, щоб продовжити дію сигналу *INTA*, що посилається на контролер переривань 8259*A*. Перший цикл дозволяє ведучому контролеру визначити, який з ведених контролерів викликав переривання. У другому циклі МП зчитує із шини даних вектор переривання, що використовується для знаходження адреси у таблиці векторів переривань. Цикл шини СТОП-ЗУПИНКА (*Halt/Shutdown*) виникає або під час виконання команди *HLT*, або під час оброблення особливого випадку 8 (табл. 3.2) у захищеному режимі.

Типи переривань. Переривання і винятки в реальному і захищеному режимах наведено у табл. 3.2.

Кожному номеру переривання відповідає елемент у таблиці *IDT* дескрипторів переривань, яка містить вектори переривань. Після скидання МП, а також під час роботи МП у

Таблиця 3.2. **Переривання і винятки МП *i*80286**

Номер переривання	Адреса повернення	Функції	
		Реальний режим	Захищений режим
0	Перший байт команди	Помилка ділення — виникає, якщо частка занадто велика або дільник дорівнює нулю	
1	Наступна команда	Переривання покрокової роботи	
2	Немає	Немасковане переривання	
3	Наступна команда	Контрольна зупинка (*INT* 3)	
4	Те саме	Особливий випадок переповнення (*INTO*)	
5	Перший байт команди	Особливий випадок виходу з діапазону — виникає за виконання команди *BOUND*	
6	Те саме	Неприпустимий код операції — виникає, якщо зустрічається недійсний код операції або команда завдовжки понад 10 байт	
7	– « –	Співпроцесор недоступний	Співпроцесор недоступний або відбулося перемикання задач

Продовж. табл. 3.2

Номер переривання	Адреса повернення	Функції	
		Реальний режим	Захищений режим
8	– « –	–	Подвійна відмова — ситуація, коли МП виявляє два незалежних винятки у процесі відпрацювання однієї команди. Якщо під час його обслуговування станеться виняток, то МП вимкнеться (цикл *Shutdown*). У цьому стані процесор припиняє свої дії і виводиться з нього сигналами скидання або немаскованого переривання
9	Невизначена	Порушення межі сегмента співпроцесором — виникає, якщо операнд співпроцесора не поміщається у сегмент, наприклад, 16-розрядний операнд має зміщення *0FFFFH*	
10	Перший байт команди	–	Неприпустимий сегмент стану задачі
11	Те саме	–	Сегмента немає
12	– « –	Порушення межі сегмента стеку	Порушення межі сегмента стеку або стеку немає
13	– « –	Порушення межі сегмента даних або коду — виникає, якщо операнд або код операції у сегмент не поміщається	Порушення захисту — виникає у таких випадках: за виходу за межі таблиці дескрипторів; з порушенням привілеїв; після завантаження недійсного дескриптора або типу сегмента; за спроби запису в сегмент коду або сегмент даних, призначених лише для читання, під час читання лише з виконуваного сегмента кодів; за спроби виконати привілейовані команди, дозволені лише для певних рівнів *CPL* і *IOPL*

Продовж. табл. 3.2

Номер переривання	Адреса повернення	Функції	
		Реальний режим	Захищений режим
14	Зарезервовано		
15	– « –		
16	Наступна команда *ESC* або *WAIT*	Особливий випадок співпроцесора — виникає в МП *i*80286 за будь-якого немаскованого особливого випадку в співпроцесорі	
17	Зарезервовано		
	⋮		
31	Зарезервовано		

реальному режимі, таблиця векторів *IDT* розміщується, починаючи з нульової адреси, однак командою завантаження регістра *IDTR* можна змінити її місцезнаходження у межах першого мегабайта і зменшити розмір таблиці.

Приклад 3.5. Визначити, чи виникне переривання у процесі роботи МП у реальному режимі під час виконання команди

MOV AX,[0FFFFH].

Під час виконання цієї команди в акумулятор *AX* пересилається слово з сегмента даних, причому молодший байт слова має зміщення *0FFFFH*, а старший — 0000*H*. Цей випадок є порушенням межі сегмента і виникає переривання 13.

3.2. Архітектура 32-розрядних мікропроцесорів

Існуючі 32-розрядні МП *i*386, *i*486, Pentium, Pentium Pro і Pentium II мають розрядність регістрів та шини адреси, яка дорівнює 32. Шина даних для процесорів *i*386, *i*486 є 32-розрядною, а для процесорів Pentium, Pentium Pro і Pentium II — 64-розрядною. Вони дають змогу адресувати 4 Гбайт пам'яті за наявності засобів підтримки сегментної та сторінкової адресації пам'яті. Процесори мають чотирирівневу систему захисту пам'яті та портів введення-виведення, можуть працювати у багатозадачному режимі. До режимів роботи МП

*i*80286 доданий Virtual Real Mode — режим віртуального процесора *i*8086. Мікропроцесори допускають паралельну роботу кількох віртуальних процесорів *i*8086 під керуванням операційної системи типу Windows, OS/2, Unix. Процесори оперують з бітами, полями бітів, 8-, 16- та 32-бітовими операндами, рядками бітів, байтів, слів (16-розрядних даних) і подвійних слів (32-розрядних даних). В архітектуру процесорів введено засоби налагодження і тестування.

Програмна модель. Програмну модель 32-розрядного процесора наведено на рис. 3.8. Вона містить такі групи регістрів:

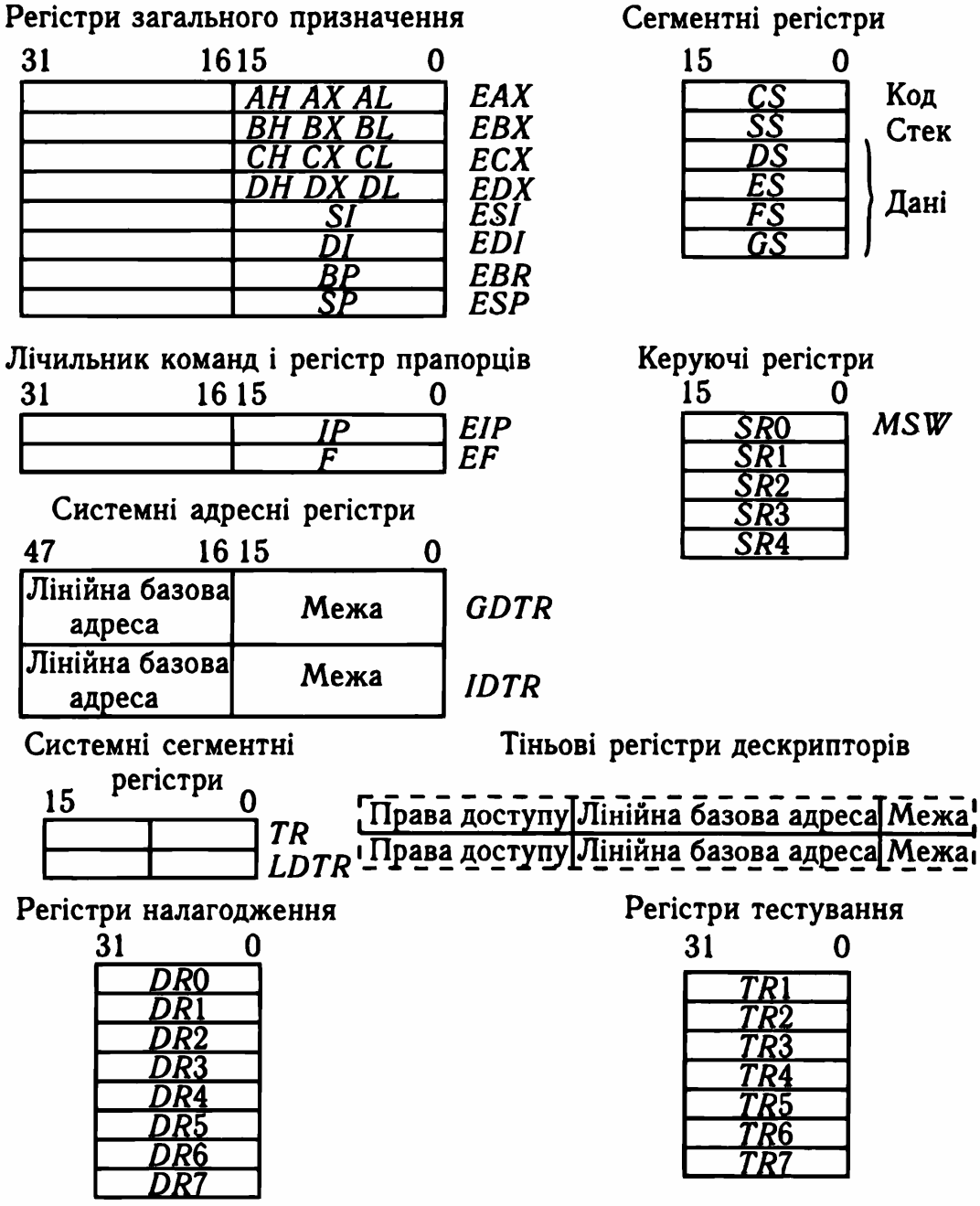


Рис. 3.8. Програмна модель 32-розрядного процесора

регістри загального призначення, лічильник команд, регістр прапорців, сегментні регістри, регістри керування, системні адресні регістри, регістри налагодження та тестування.

Регістри загального призначення містять усі регістри даних і регістри-покажчики МП *i*8086 та *i*80286 і стільки само додаткових 32-розрядних регістрів. У позначенні 32-розрядних регістрів використовується літера *E* (*Expanded* — *розширений*).

Лічильник команд EIP містить зміщення наступної виконуваної команди в сегменті кодів. Для 16-розрядних адрес використовуються молодші 16 розрядів (*IP*).

Регістр прапорців EF розширено до 32 розрядів. Молодші 16 розрядів регістра *EF* створюють регістр прапорців *F* 16-розрядного процесора. У регістр *EF* додано нові прапорці:

ID (*Identification Flag*) — прапорець дозволу команди ідентифікації *CPUID* (Pentium+ і деякі процесори типу 486)*;

VIP (*Virtual Interrupt Pending*) — віртуальний запит переривання (Pentium+);

VIF (*Virtual Interrupt Flag*) — віртуальна версія прапорця дозволу переривання *IF* для багатозадачних систем (Pentium+);

AC (*Alignment Check*) — прапорець контролю вирівнювання. Використовується лише на рівні привілеїв 3. Якщо *AC* = 1 і *AM* = 1 (*AM* — біт у регістрі керування *CR*0), то у разі звернення до операнда, не вирівняного за відповідною межею (2, 4, 8 байт)**, відбувається виняток 17 (*i*486+);

VM (*Virtual* 8086 *Mode*) — у захищеному режимі вмикає режим віртуального процесора 8086. Спроба використання привілейованих команд у цьому випадку приведе до винятку 13;

RF (*Resume Flag*) — прапорець поновлення. У режимі налагодження одиничне значення *RF* дозволяє здійснити рестарт команди після особливого випадку налагодження.

Сегментні регістри. Крім сегментних регістрів МП *i*8086 та *i*80286 (*DS*, *CS*, *SS*, *ES*), програмна модель містить два додаткових сегментних регістри даних — *FS* і *GS*. З кожним з шести сегментних регістрів пов'язані тіньові регістри дескрип-

---

*У подальшому позначення *i*386+, *i*486+, Pentium+ означають, що наведені дані справедливі для вказаної моделі МП і всіх старших моделей.

**Вирівнювання операнда по межі 2, 4, 8 означає, що адреса операнда є кратною, тобто 2, 4, 8.

торів. У тіньові регістри у захищеному режимі переписуються 32-розрядна базова адреса сегмента, 20-розрядна межа й атрибути (права доступу) з дескрипторних таблиць.

Керуючі регістри *CR*0—*CR*3 (*Control Register*) зберігають ознаки стану процесора, спільні для всіх задач. Молодші чотири розряди регістра *CR*0 містять біти регістра *MSW* МП *i*80286 і деякі інші біти керування. Регістр *CR*1 зарезервовано; регістр *CR*2 зберігає 32-розрядну лінійну адресу, за якою отримано відмову сторінки пам'яті; регістр *CR*3 у старших 20 розрядах зберігає фізичну базову адресу таблиці каталога сторінок і біти керування кеш-пам'яттю, а регістр *CR*4 (Pentium+) містить біти дозволів архітектурних розширень МП.

Системні адресні регістри. Системні покажчики (регістри глобальної дескрипторної таблиці *GDTR* і таблиці переривань *IDTR*) зберігають відповідно 32-розрядні базові адреси і 16-розрядні межі таблиць. Системні сегментні регістри задач *TR* і локальної дескрипторної таблиці *LDTR* є 6-розрядними селекторами. Їм відповідають тіньові регістри дескрипторів, які містять 32-розрядну базову адресу сегмента, 20-розрядну межу і права доступу.

Регістри налагодження *DR*0—*DR*3 (*Debug Register*) зберігають 32-розрядні адреси точок зупину в режимі налагодження, *DR*4—*DR*5 зарезервовані і не використовуються; *DR*6 — відображує стан контрольної точки; *DR*7 — керує розміщенням у програмі контрольних точок.

Регістри тестування *TR* (*Test Register*) входять до групи модельно-специфічних регістрів, їхній склад і кількість залежать від типу процесора: в МП 386 використовувалися два регістри — *TR*6 і *TR*7, у Pentium-12 — *TR*1—*TR*12. Ця група регістрів зберігає результати тестування МП і кеш-пам'яті.

Сегментна організація пам'яті. У 32-розрядних МП розрізняють три адресні простори пам'яті — логічний, лінійний і фізичний. Логічна адреса (або віртуальна) складається з селектора і зміщення *EA*. Лінійна адреса утворюється додаванням базової адреси сегмента до ефективної адреси. Фізична адреса пам'яті створюється після перетворення лінійної адреси блоком сторінкової переадресації.

Організація пам'яті залежить від режиму роботи МП. У *реальному* і *віртуальному* режимах *i*8086 адресація пам'яті така сама, як у МП *i*8086. У *захищеному* режимі здійснюється сегментна і сторінкова організація пам'яті. Сегментна організація використовується на прикладному рівні, а сторінкова — на системному. Формування адреси комірки пам'яті

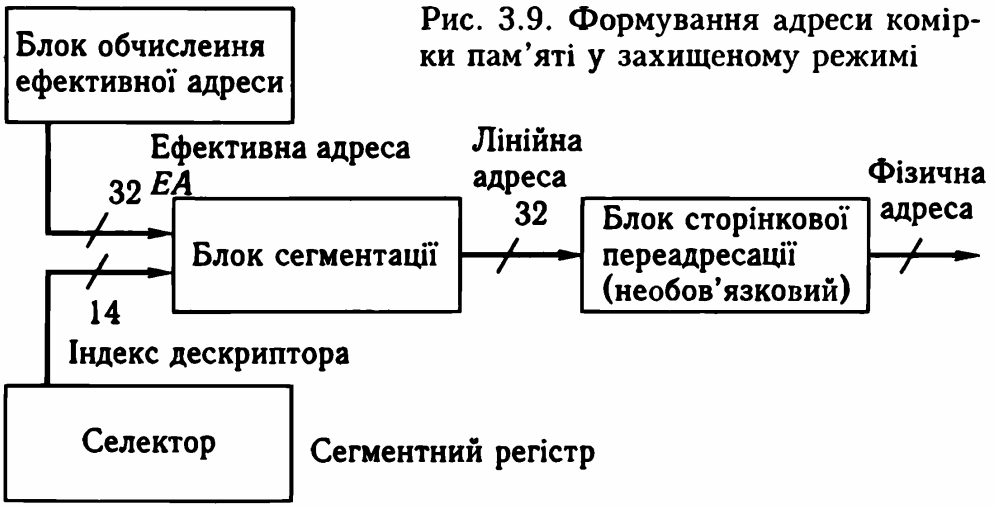


Рис. 3.9. Формування адреси комірки пам'яті у захищеному режимі

у захищеному режимі подано на рис. 3.9. Блок сегментації перетворює простір логічних адрес на простір лінійних адрес. Вихідними даними для блока сегментації є зміщення *EA* у сегменті та сегментний регістр, які задаються у команді. Вміст сегментного регістра у захищеному режимі є селектором. Він містить інформацію про тип дескрипторної таблиці (глобальної або локальної) та індекс дескриптора (див. рис. 3.1). Індекс дескриптора є номером дескриптора у таблиці. Дескриптор містить базову адресу сегмента. Лінійна адреса створюється додаванням базової та ефективної адрес згідно з рис. 3.2.

Блок сторінкової переадресації формує фізичну адресу пам'яті. За вимкненого блока лінійна адреса збігається з

Таблиця 3.3. **Типи адресації у 32-розрядних процесорах**

Тип адресації	Обчислення *EA*
Регістрова	*EA* = вмісту РЗП
Пряма	*EA* = *Disp*
Непряма регістрова	*EA* = *Base*
Базова	*EA* = *Base* + *Disp*
Індексна	*EA* = *Index* + *Disp*
Масштабована індексна	*EA* = *Scale* × *Index* + *Disp*
Базова індексна	*EA* = *Base* + *Index*
Масштабована базова індексна	*EA* = *Base* + *Index* × *Scale*
Базова індексна зі зміщенням	*EA*= *Base* + *Index* +*Disp*
Масштабована базова індексна зі зміщенням	*EA* = *Base* + *Index* × *Scale* + *Disp*

фізичною. Блок обчислення ефективної адреси обчислює адресу-зміщення операнда у сегменті за одним з наступних типів адресації, наведених у табл. 3.3. Алгоритм обчислення адреси комірки пам'яті для різних типів адресації показано на рис. 3.10. Регістри загального призначення МП можуть виконувати функції таких регістрів: базового *Base*, індексного *Index*, масштабування множника *Scale* і зміщення *Disp*. У табл. 3.4 подано використання цих регістрів залежно від режимів адресації: 16- або 32-розрядної. Масштабовані типи адресації можливі лише у 32-розрядному режимі адресації.

У реальному режимі за замовчуванням використовується 16-бітова адресація, але за допомогою префікса зміни розрядності адреси можна перемкнути на 16-розрядний режим. У захищеному режимі тип адресації залежить від біта *D* у дескрипторі кодового сегмента (за $D = 0$ використовується 16-розрядна адресація, а за $D = 1$ — 32-розрядна).

Використання сегментних регістрів під час адресації пам'яті визначається типом звернення до пам'яті (табл. 3.5). Для деяких типів звернень допускається заміна сегментного ре-

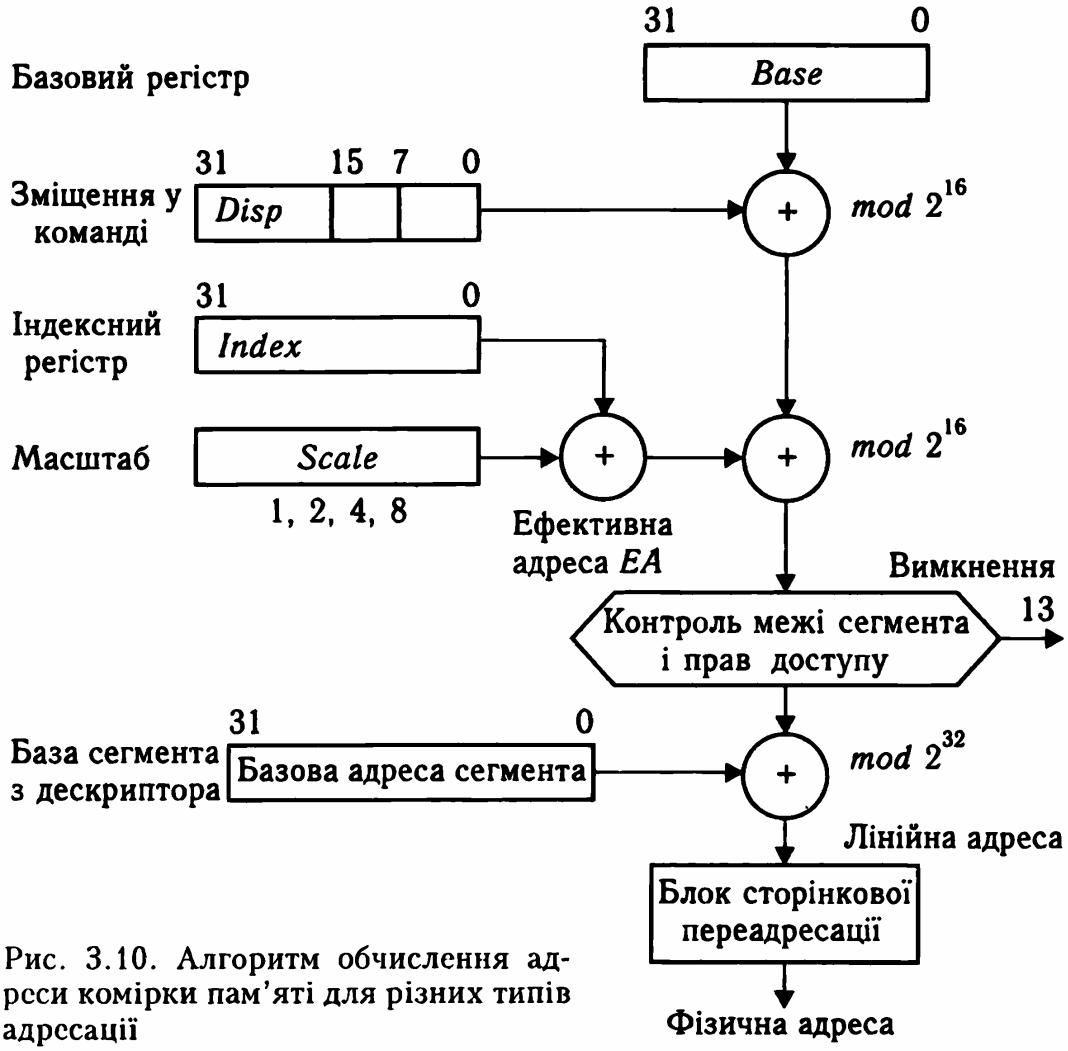


Рис. 3.10. Алгоритм обчислення адреси комірки пам'яті для різних типів адресації

Таблиця 3.4. **Використання РЗП під час обчислення ефективної адреси**

Компонент	Адресація	
	16-розрядна	32-розрядна
Базовий регістр (*Base*)	*BX* або *BP*	Будь-який 32-розрядний РЗП
Індексний регістр (*Index*)	*SI* або *DI*	Будь-який 32-розрядний РЗП, крім *ESP*
Масштаб (*Scale*)	1	1, 2, 4 або 8
Зміщення (*Disp*)	0, 8 або 16 біт	0, 8 або 32 біт

Таблиця 3.5. **Використання сегментних регістрів для адресації пам'яті**

Тип звернення до пам'яті	Сегментний регістр		Зміщення
	за замовчуванням	альтернативний	
Вибірка команд	*CS*	Немає	*IP*, *EIP*
Стекові операції	*SS*	Немає	*SP*, *ESP*
Адресація змінної	*DS*	*CS*, *ES*, *SS*, *FS*, *GS*	*EA*
Рядок-джерело	*DS*	*CS*, *ES*, *SS*, *FS*, *GS*	*SI*
Рядок-приймач	*ES*	Немає	*DI*
Використання *BP*, *EBP* або *ESP* як базового регістра	*SS*	*CS*, *ES*, *DS*, *FS*, *GS*	*EA*

гістра, що вводиться, застосуванням префіксів команд *CS*:, *DS*:, *SS*:, *ES*:, *FS*:, *GS*:, наприклад

$$ADD\ FS{:}[ESI],EAX; \quad [FS{:}ESI] \leftarrow [FS{:}ESI]+EAX.$$

Приклад 3.6. Знайти значення фізичної адреси операнда в команді пересилання у регістр *AL* вмісту комірки пам'яті

$$MOV\ AL, [BX + 4 \cdot SI + 1000H],$$

якщо базова адреса сегмента даних дорівнює $12\,456\,789H$, а вміст регістрів $BX = 0120H$, $SI = 1234H$. Блока сторінкової переадресації немає.

Ефективна адреса комірки пам'яті

$$EA = 0120H + 4 \cdot 1234H + 1000H = 59F0H.$$

Виконавши операцію додавання 32-розрядної базової адреси з ефективною адресою *EA*, отримаємо лінійну адресу, яка збігається і з фізичною адресою:

$$12456789H + 59F0H = 1245C179H.$$

Отже, фізична адреса дорівнює $1245C179H$.

Формування базової адреси сегмента пояснює рис. 3.11. Поле *TI* селектора сегмента визначає робочу дескрипторну таблицю (глобальну або локальну), де знаходиться початкова адреса сегмента.

Поле *RPL* визначає запрошений рівень привілею сегмента, а поле *Index* — зміщення від початкової адреси таблиці. Зазначимо, що початкова адреса таблиці зберігається або у регістрі *GDTR* — для глобальної таблиці, або у тіньовому регістрі. В останньому випадку регістр *LDTR*, в свою чергу, є селектором і вказує, де в глобальній дескрипторній таблиці знаходиться інформація про початкову адресу локальної таблиці. Ця інформація переписується у тіньовий регістр.

Формат дескриптора для 32-розрядних процесорів подано на рис. 3.12. Дескриптор МП *i*80286 містить нуль у бітах 63—48, а поля базової адреси і межі займають відповідно 24 і 16 біт. У 32-розрядному МП поле базової адреси займають другий, третій, четвертий і сьомий байти дескриптора. У процесі виконання команди ці байти об'єднуються в одну 32-розрядну базову адресу.

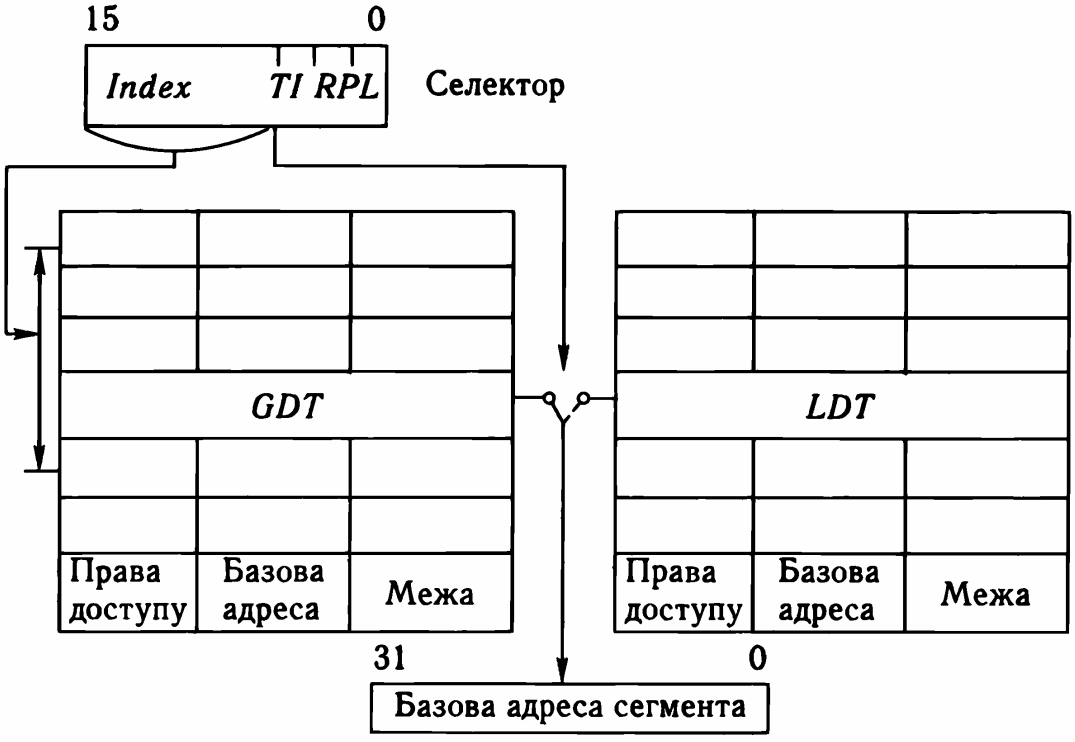


Рис. 3.11. Формування базової адреси сегмента

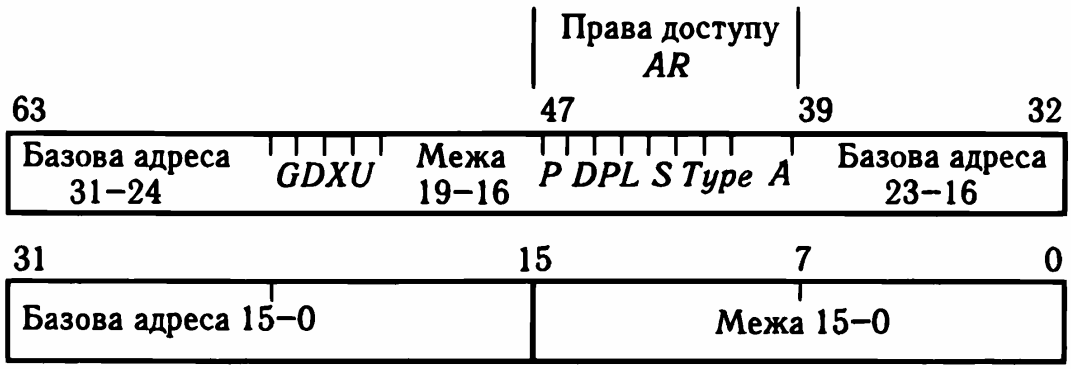


Рис. 3.12. Формат дескриптора для 32-розрядних процесорів

Поле межі займає байти з номерами 0, 1 і молодші чотири біти шостого байта дескриптора. Межа задає максимальне зміщення у сегменті або останню одиницю, що адресується в сегменті. За 20-розрядної межі максимальне значення елементів, що адресуються, становить 2^{20}. Оскільки елементом сегмента є не лише байт, а й сторінка 4 Кбайт, сегмент може містити від одного байта до 4 Гбайт. Байт з номером 5 дескриптора *AR* (*Access Rights*) містить право доступу, зокрема, такі біти керування: *P* (*Present*) — біт наявності; *DPL* (*Descripter Privilege Level*) — поле рівня привілеїв сегмента; *S* (*System*) — системний біт; *Type* — поле типу сегмента; *A* (*Accessed*) — біт звернення.

Біт присутності *P* дорівнює одиниці, якщо сегмент знаходиться у фізичній пам'яті (ОЗП). У системі віртуальної пам'яті операційна система може передавати вміст деяких сегментів на диск, при цьому вона скидає біт *P* у стан логічного нуля в дескрипторі цього сегмента. Якщо програма після цього знову звертається до сегмента, виникає особливий випадок відсутності сегмента. Операційна система шукає вільну область фізичної пам'яті (при цьому, можливо, відправляє на диск деякий інший сегмент), копіює вміст запрошеного сегмента з диска у пам'ять, записує в його дескриптор нову базову адресу та здійснює рестарт команди, що викликала особливий випадок 11 відсутності сегмента. Описаний процес називають *свопінгом* (*swapping*), або *довантаженням*.

Поле рівня привілеїв сегмента *DPL* містить 2 біти. Найвищому рівню привілею відповідає значення 0, найнижчому — значення 3.

Системний біт S має нульове значення ($S = 0$) у дескрипторах сегмента кодів, системних сегментів для зберігання локальних таблиць дескрипторів, станів задач *TSS* (*Task State Segment*) та у дескрипторах, що називають *вентилями* (*Gate*), або *шлюзами*. В інших випадках $S = 1$.

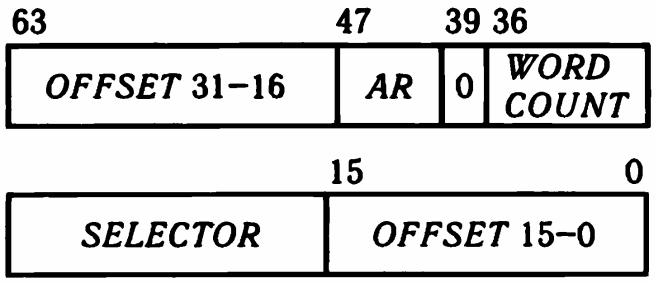


Рис. 3.13. Формат вентилів

Вентиль містить інформацію про логічну адресу входу до деякої системної програми і займає 8 байт. Формат вентилів показано на рис. 3.13.

Вентилі призначені для передавання керування і містять логічну адресу переходу у вигляді селектора *SELECTOR* і 32-розрядного зміщення *OFFSET*. *Вентилі виклику* використовують для викликів процедур зі зміною рівня привілеїв, *вентилі задач* — для перемикання задач, *вентилі переривань* та *вентилі пастки* — для переходу до процедур обслуговування переривань, при цьому вентилі переривань забороняють переривання (відбувається скидання прапорця *IF*), а вентилі пастки — не забороняють.

Поле *WORD COUNT* (див. рис. 3.13) у вентилях виклику визначає кількість слів, які копіються зі стеку однієї процедури у стек іншої процедури. Для інших вентилів поле *WORD COUNT* містить нульові значення.

Поле типу сегмента *Type* займає три розряди і визначає тип сегмента згідно з табл. 3.6.

Біт звернення *A*. У сегментах коду і даних *A* = 0 означає, що до сегмента не було звернень. У системних об'єктах поле *Type* разом з бітом *A* визначає тип системного об'єкта (див. табл. 3.6).

У старшій тетраді шостого байта дескриптора знаходяться такі біти керування:

G (*Granularity*) — біт гранулярності. Якщо *G* = 0, одиницею пам'яті в сегменті є байт, а якщо *G* = 1, — сторінка завдовжки 4 Кбайт;

D (*Default size*) — біт розміру. Якщо *D* = 0, операнди в пам'яті вважають 16-розрядними, а якщо *D* = 1, — 32-розрядними. Застосовується для сумісності з МП *i*80286;

U (*User*) — біт користувача. Може бути встановлений або відбутися скидання програмно.

Як видно з опису дескриптора, 32-розрядний МП дозволяє створювати сегменти, в яких можуть виконуватися операції зчитування, читання-записування, виконання або виконання-

Таблиця 3.6. **Типи сегментів і системних об'єктів**

Type	*A*	Тип сегмента	Дозволені операції
Сегменти кодів і даних			
0 0 0	*A*	Даних	Тільки зчитування
0 0 1	*A*	–«–	Зчитування і запис
0 1 0	*A*	Стеку	Тільки зчитування
0 1	*A*	–«–	Зчитування і запис
1 0 0	*A*	Коду	Тільки виконання
1 0 1	*A*	–«–	Виконання і зчитування
1 1 0	*A*	Підлеглий сегмент коду**	Тільки виконання
1 1 1	*A*	–«–	Виконання і зчитування
Системні сегменти*			
0 0 0	1	Доступний сегмент *TSS* стану задачі* *i*80286	
0 0 1	0	Таблиця локальних дескрипторів *LDT*	
0 0 1	1	Зайнятий сегмент *TSS* стану задачі *i*80286	
1 0 0	1	Доступний сегмент *TSS* стану задачі *i*386+	
1 0 1	0	Зарезервовано	
1 0 1	1	Зайнятий сегмент *TSS* стану задачі* *i*386+	
Вентилі*			
0 1 0	0	Вентиль виклику *i*80286 (*Call Gate*)	
0 1 0	1	Вентиль задачі *i*80286 (*Task Gate*)	
0 1 1	0	Вентиль переривання *i*80286 (*Interrupt Gate*)	
0 1 1	1	Вентиль пастки *i*80286 (*Trap Gate*)	
1 1 0	0	Вентиль виклику *i*386 + (*Call Gate*)	
1 1 0	1	Вентиль задачі *i*386 +(*Task Gate*)	
1 1 1	0	Вентиль переривання *i*386 + (*Interrupt Gate*)	
1 1 1	1	Вентиль пастки *i*386 + (*Trap Gate*)	

Примітки: * На практиці такі сегменти стеку не використовуються.
** Підлеглі сегменти кодів подано у п. 3.2.4.
*** Інші стани полів *Type* та *A* не використовуються.

зчитування. Для створення характерних для МП *i*8086 сегментів, у яких виконуються одночасно всі перелічені операції, використовують перекриття сегментів пам'яті, тобто початкова адреса одного сегмента є адресою іншого.

Сторінкова організація пам'яті. Цей тип організації, здебільшого застосовують у системах віртуальної пам'яті, що

дає змогу програмісту використовувати більший простір адрес, ніж існуюча фізична пам'ять. Враховуючи властивість просторової локальності кодів і даних (близького розміщення необхідних комірок пам'яті), доцільно оперувати не байтами, а деякими невеликими модулями пам'яті — сторінками. За сторінкового перетворення весь лінійний адресний простір 32-розрядного МП ємністю 4 Гбайт розбивається на 2^{20} сторінок по 4 Кбайт. Фізичний простір пам'яті мікропроцесорної системи також розбивається на сторінки, причому у фізичній пам'яті сторінок значно менше 2^{20}. Наприклад, за ємності пам'яті 4 Мбайт кількість фізичних сторінок (їх ще називають сторінковими кадрами, або *page frame*) становить 2^{10}. Відсутні у ВІС фізичної пам'яті зберігаються у зовнішній пам'яті (накопичувачі на твердому магнітному диску) і за потреби завантажуються у фізичну пам'ять, тобто відбувається процес свопінгу. Прикладні програми можуть розпоряджатися всім простором віртуальної пам'яті — 4 Гбайт. Процес сторінкового перетворення адреси подано на рис. 3.14.

У процесі перетворення старші 20 біт 32-розрядної лінійної адреси замінюються іншим 20-розрядним значенням — номером фізичної сторінки згідно з механізмом перетворення

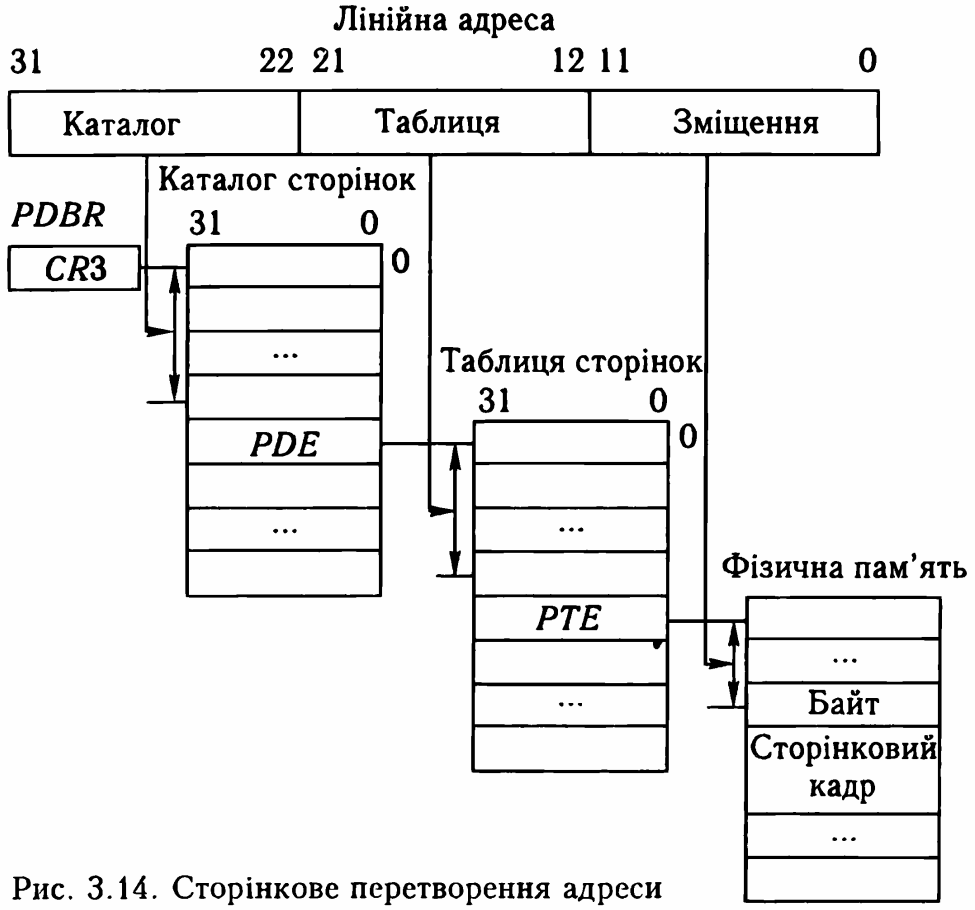


Рис. 3.14. Сторінкове перетворення адреси

адреси (див. рис. 3.14). Регістр керування *CR3 PDBR* (*Page Directory Base Register*) (див. п. 3.2) містить фізичну базову адресу каталога сторінок. *Каталог сторінок* знаходиться у фізичній пам'яті постійно і не бере участі у свопінгу. Він містить 1024 32-розрядні адреси *PDE* — (*Page Directory Entry*). Кожна з них є початковою адресою таблиць сторінок. *Таблиця сторінок PTE* (*Page Table Entry*) містить адреси сторінкових кадрів у фізичній пам'яті.

Фізична базова адреса каталога сторінок формується із значення рядка таблиці сторінок *PTE* і 12 розрядів зміщення лінійної адреси.

Захист за привілеями. Систему привілеїв призначено для запобігання недозволеним взаємодіям користувачів, несанкціонованому доступу до даних, пошкодженню програм і даних. Частково ці задачі розв'язуються організацією захищеного режиму пам'яті, частково — захистом за привілеями; 32-розрядні процесори підтримують чотири рівні привілеїв 0—3, причому рівень 0 є найбільш привілейованим. Рівень 0 зазвичай присвоюється ядру операційної системи, рівень 1 — системним сервісам, рівень 2 — розширенням операційної системи і рівень 3 — прикладним програмам користувача. Під час виконання програми контролюється, чи може програма:

- здійснювати привілейовані команди;
- звертатися до даних інших програм;
- передавати керування іншій програмі командами передавання керування типу *FAR*.

До привілейованих команд належать команди, що змінюють сегментацію, впливають на механізм захисту, модифікують прапорець дозволу переривань *IF*. За спроби виконати ці команди на рівнях привілеїв 1, 2, 3 генерується виняток 13.

Для контролю звернення програми до даних інших програм використовуються поля *CPL* (*Current Privilege Level* або *Code Privilege Level* — поточний рівень привілеїв; задається полем *RPL* селектора *CS*) і дескриптора даних *DPL*. Доступ до даних дозволяється якщо $CPL \leq DPL$.

Передавання керування програм різних рівнів привілеїв здійснюється за допомогою використання:

- підлеглих сегментів коду;
- дескрипторів вентилів викликів (шлюзів).

У підлеглих сегментах виконання команд можливе, якщо поточний рівень привілеїв (*CPL*) не нижчий від рівня привілеїв дескриптора (*DPL*) підлеглого сегмента, у непідлеглих — керування сегмента передається за $CPL = DPL$. Зазвичай у підлеглих сегментах кодів розміщують бібліотеки,

до яких можуть звертатися програми різних рівнів привілеїв. Використання підлеглих кодових сегментів не змінює поточний рівень привілеїв. Єдиним способом зміни рівня привілею є використання вентилів викликів. Вентилі ідентифікують дозволені точки входу в кодові сегменти з більшим рівнем привілею. У дескрипторі вентиля задається повна адреса точки входу (селектор: зміщення) тієї процедури, якій передається керування.

Перемикання задач. У багатозадачних системах і системах з великою кількістю користувачів МП виконує деяку частину команд однієї задачі (програм), після цього перемикається на виконання іншої задачі і так триває доти, доки знову не повертається до першої задачі. Для підтримки багатозадачного режиму в МП є такі засоби:

- сегмент стану задачі *TSS*;
- дескриптор сегмента стану задачі;
- регістр задачі *TR*;
- вентиль задачі.

Дескриптор сегмента стану задачі вказує на сегмент, що містить повний стан задачі, а вентиль задачі містить селектор, що вказує на дескриптор *TSS*. Регістр *TR* є селектором сегмента *TSS* поточної задачі. Кожна задача має свій сегмент стану. В сегменті *TSS* міститься інформація про стан процесора на час перемикання задач — вміст майже всіх регістрів МП, включаючи регістр прапорців, роздільні покажчики стеків для рівнів привілеїв 0, 1, 2 і посилання на селектор *TSS* задачі, що її викликала.

Перемикання задач здійснюється або за командами міжсегментних переходів *JMP FAR* чи викликів підпрограм *CALL FAR*, або за апаратними чи програмними перериваннями і винятками. У першому випадку програма має посилатися на сегмент стану задачі *TSS* або на дескриптори вентиля задачі в *GDT*(*LDT*), у другому — відповідний перериванню дескриптор в таблиці переривань *IDT* має бути дескриптором вентиля задачі.

Під час передавання керування викликаній задачі за командою *IRET* перевіряється прапорець вкладеної задачі *NT* (*Nested Task*). Якщо $NT = 0$, команда *IRET* працює у звичайному режимі, залишаючись у поточній задачі. Якщо $NT = 1$, команда *IRET* виконує перемикання на попередню задачу.

3.3 Особливості архітектури мікропроцесорів *i*386 та *i*486

Мікропроцесор *i*386. Перший 32-розрядний процесор *i*386 було створено у 1985 р. Він виконаний за технологією 1,5 мкм і містить 275 тис. транзисторів. Розрядність регістрів, шини даних та адреси дорівнює 32. Ємність пам'яті, що прямо адресується, становить 4 Гбайт. Процесор може працювати у трьох режимах — реальному, захищеному і віртуального процесора 8086 — V86. Допускається паралельна робота кількох віртуальних процесорів 8086 під керуванням операційної системи типу Windows, OS/2, Unix. Перемикання режимів відбувається швидше, ніж у МП *i*80286. Є механізми сторінкової адресації, які істотно підвищують ефективність роботи з пам'яттю понад 1 Мбайт навіть у межах *DOS*. Черга команд — 16 байт. Мікропроцесор *i*80386 має модифікації: *DX* — регістри, шини даних та адреси є 32-розрядними; *SX* — із зовнішньою 16-розрядною шиною даних та 24-розрядною шиною адреси; *SL* — відрізняється від модифікації *SX* зниженим енергоспоживанням та вбудованим контролером зовнішньої кеш-пам'яті на 16—64 Кбайт. До комплекту 386*SL* входить мікросхема 82360*SL* — набір периферійних контролерів для ноутбуків. Основні характеристики процесорів 80 × 86 наведено у табл. 3.7.

Мікропроцесор *i*486. Цей процесор випустили у 1989 р. Він характеризується значно вищою швидкодією порівняно з мікропроцесором *i*8086. Мікропроцесор виконано за технологією 1 мкм; містить 1,2 млн транзисторів. Основні особливості МП *i*486 — наявність внутрішньої кеш-пам'яті, вбудованого математичного співпроцесора, сумісного за командами арифметичного співпроцесора *i*387. У МП *i*486 збільшено чергу команд, прискорено виконання операцій як у цілочисловому АЛП, так і в блоці математичного співпроцесора за рахунок архітектури, введено множення тактової частоти системної плати. У модифікаціях процесора 486DX2 внутрішня частота дорівнює подвоєній зовнішній, а в процесорах 486DX4 кратність може бути 2, 2,5, 3. Математичного співпроцесора у модифікаціях SX та в деяких модифікаціях SL немає. Залежно від модифікації процесори DX4 можуть працювати від джерела живлення 5 В та 3,3 В і мають режим *SMM* (*System Management Mode*), що дає змогу керувати енергоспоживанням.

Крім розглянутих вище процесорів фірми Intel (див. табл. 3.7) є аналогічні за технічними характеристикам процесори, які

Таблиця 3.7. **Основні характеристики процесорів 80 × 86 фірм Intel і IBM**

Процесор	Розрядність			Ємність кеш-пам'яті, Кбайт	Наявність співпроцесора	Частота процесора, МГц
	регістрів	шини даних	шини адреси			
8088	16	8	20	–	–	4,77–8
8086	16	16	20	–	–	5
286	16	16	24	–	–	6–25
386SX	32	16	24	–	–	16–23
386SL	32	16	24	–	–	25
386SLC	32	16	24	8	–	25–40
486SLC	32	16	24	16	–	25–40
486SLC2	32	16	24	16	–	40–66
486SLC3	32	16	24	16	–	75
386DX	32	32	32	–	–	25–40
486DLC	32	32	32	16	–	25–40
486SX	32	32	32	8	–	16–33
486BL2	32	32	32	16	–	40–66
486BL3	32	32	32	16	–	75–100
486DX	32	32	32	8	+	25–50
487SX	32	32	32	8	+	25–50
486SL	32	32	32	8	+	25–50
486DX2	32	32	32	8/16	+	40–80
486DX4	32	32	32	16	+	75–120

виготовляють фірми IBM, AMD, Cyrix, Texas Instruments. Так, процесор 386SLC — це поліпшений варіант процесора 386SX. Він має внутрішню кеш-пам'ять і характеризується прискореним виконанням операцій. Мікропроцесор 486SLC є варіантом процесора *i*486SX, а процесори SLC2/SLC3 здійснюють подвоєння (потроєння) зовнішньої частоти.

Процесори SCL2 і SL характеризуються зниженим енергоспоживанням і мають напругу живлення 3,3 В. Так, більшість процесорів фірми AMD мають знижені енергоспоживання (літера L у позначенні *BIC*) і напругу живлення 3,3 В (літера V у позначенні МП), наприклад, процесор AMD 5X-133ADV — це варіант мікропроцесора *i*486 зі збільшеною частотою вчетверо та зниженою напругою живлення.

Внутрішня кеш-пам'ять. Починаючи з МП *i*486, застосовується внутрішнє роздільне видавання команд і даних (докладно принципи організації кеш-пам'яті розглянуто в п. 16.5). Якщо область, яка адресується, відображено у кеш-пам'яті (випадок попадання — *cache hit*), то запит на читання обслуговується лише кеш-пам'яттю без звернення до основної пам'яті. У разі запиту на запис спочатку модифікується інформація у кеш-пам'яті, а після цього, залежно від типу кеш-пам'яті, й основна пам'ять.

У перших процесорах *i*486 використовувався режим наскрізного запису *Write Through*, коли інформація одночасно записувалася як у буфер, так і в ОЗП. У новіших модифікаціях використано режим зворотного запису *Write Back*, який полягає у тому, що копія блока записується в ОЗП лише тоді, коли вміст його змінювався.

Заповнення рядка кеш-пам'яті процесор намагається виконати найшвидше — пакетним циклом з 32-бітовими передачами. Вибір рядка для заміщення новими даними здійснюється на підставі аналізу біта *LRU* (*Least Recently Used*), який виконується лише для кеш-промахів читання. У разі кеш-промахів запису заповнення рядків не відбувається.

Пакетний режим передачі даних. Пакетний режим передавання даних (*Burst Mode*) призначений для швидких операцій з рядками кеш-пам'яті. При цьому вміст чотирьох 32-розрядних комірок основної пам'яті пересилається в один рядок кеш-пам'яті або навпаки — вміст одного рядка кеш-пам'яті пересилається у чотири 32-розрядні комірки основної пам'яті. Оскільки рядок кеш-пам'яті процесора 486 має довжину 16 байт (128 біт), то для його пересилання треба чотири 32-розрядних циклів шини, впродовж кожного з яких відбувається пересилання вмісту однієї 32-розрядної комірки з основної пам'яті або 32 біт з кеш-пам'яті.

Розглянемо випадок пересилання вмісту чотирьох 32-розрядних комірок основної пам'яті в один рядок кеш-пам'яті (рис. 3.15). Під час зчитування вмісту 32-розрядної комірки розряди $A0$ і $A1$ участі у формуванні адрес не беруть, оскільки вони означають положення кожного з чотирьох байтів у 32-розрядній комірці. Тому на рис. 3.15 подано значення розрядів $A31-A2$.

У першому такті $T1$ встановлюється адреса $A31-A2$, сигнали ідентифікації типу циклу і формується строб *ADS#*. Цей такт виконується процесором так само, як і за звичайного передавання даних.

У наступному такті ($T2$) передається перше 32-розрядне слово. При цьому сигнал *BLAST#* має значення *H*-рівня.

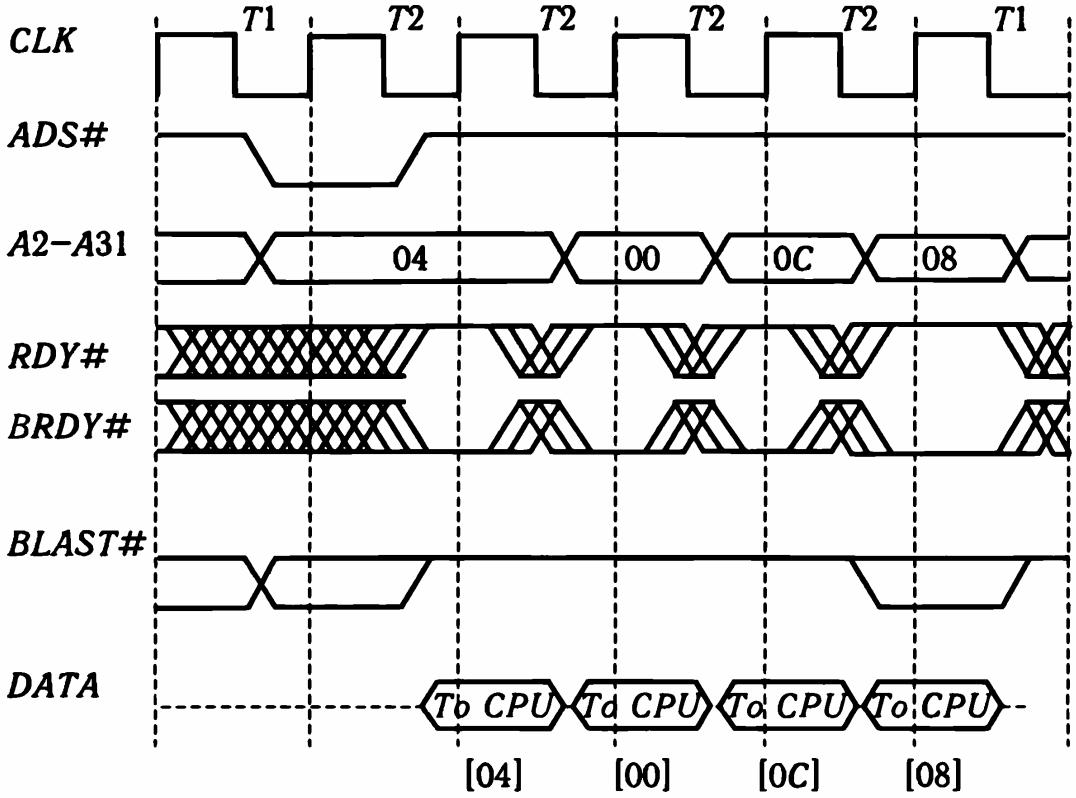


Рис. 3.15. Діаграма пересилання вмісту чотирьох 32-розрядних комірок пам'яті

Про завершення пакетного циклу процесор повідомляє зовнішньому пристрою сигналом *BLAST#*, що набуває значення *L*-рівня в останньому такті передавання пакета. Якщо зовнішній пристрій підтримує пакетний режим, він генерує сигнал готовності до пакетного передавання *BRDY#* (замість сигналу *RDY#*). У цьому разі процесор продовжує цикл як пакетний, не вводячи такти *T*1 адресації-ідентифікації (з сигналом *ADS#*), а відразу переходячи до передавання наступного 32-розрядного слова. Формуванням сигналу *RDY#* замість *BRDY#* зовнішній пристрій може у будь-який момент перервати пакетне передавання, і процесор продовжить її звичайними циклами. Для передавання 16 байт у пакетному режимі потрібно п'ять тактів шини (без тактів чекання) замість восьми тактів за звичайного режиму передавання.

Пояснимо чередування адрес 32-розрядних слів у пакетному режимі. Подамо адресу слова у такому вигляді:

*A*31–*A*4	*A*3	*A*2	*A*1	*A*0
Довільні значення	Номер 32-розрядного слова у пакетній посилці		Номер байта у 32-розрядному слові	

Розряди *A*1 та *A*0 у цій адресі дорівнюють нулю. Розряди *A*3, *A*2 визначають положення кожної з чотирьох 32-розрядних комірок у рядку кеш-пам'яті завдовжки 128 біт. Розряди *A*31—*A*4 визначають адресу 32-розрядного слова в основній пам'яті, яке пересилається у кеш-пам'ять.

Рядок кеш-пам'яті вирівняно по межі 128-розрядних даних, його адреса має нульові значення розрядів A3—A0. У пакетному передаванні завжди зчитуються дані, що відповідають одному рядку кеш-пам'яті, адреса якого кратна 128. Адреса першого 32-розрядного слова пакетної посилки кратною 128 може не бути. За нульових значень розрядів *A*31—*A*4 та *A*1, *A*0 адреса першого слова визначається розрядами *A*3, *A*2:

*A*31—*A*4	*A*3	*A*2	*A*1	*A*0	Адреса першого слова у пакетному режимі
0...0	0	0	0	0	0
0...0	0	1	0	0	4
0...0	1	0	0	0	8
0...0	1	1	0	0	*C*

Чередування адрес 32-розрядних слів у пакетному режимі залежно від адреси першого слова наведено у табл. 3.8.

Якщо перша адреса блока не збігається з межею рядка, то вона дорівнює 04*H* (див. рис. 3.15). Порядок чергування адрес при цьому відповідає другому рядку табл. 3.8, тобто дорівнює 04*H*, 00*H*, 0*CH*, 08*H*. Наведений порядок чередування адрес у пакетному режимі характерний для всіх процесорів Intel та сумісних з ними, починаючи з *i*486.

Буфери відкладеного запису призначені для запам'ятовування даних у буфері тоді, коли зовнішня шина зайнята. Процесор *i*486 має чотири буфери відкладеного запису. Інформація у буфер записується за один такт. Після закінчення поточного циклу шини інформація з буферів передається зовні — у пам'ять або ПВВ. Зовнішні операції запису з буферів виконуються у тому самому порядку, в якому

Таблиця 3.8. **Послідовність зміни адрес у пакетному циклі**

Адреса слова			
першого	другого	третього	четвертого
0	4	8	*C*
4	0	*C*	8
8	*C*	0	4
C	8	4	0

надійшли запити на запис. Якщо у незвільнених буферах усі запити на запис у пам'ять пов'язані з кеш-попаданнями, а запит на читання пов'язаний з кеш-промахом, то операція читання може відбутися раніше, ніж операції запису. Однак більше ніж один раз змінювати порядок виконання команд у МП не дозволяється, оскільки нові прочитані дані можуть замінити модифікований рядок кеш-пам'яті, з якого поновлена інформація очікує у буфері черги на запис в основну пам'ять. У такому разі друга спроба зміни послідовності команд може порушити цілісність даних. Для операцій введення-виведення зміна послідовності команд не допускається, бо це може призвести до порушення протоколу обміну.

OverDrive-процесори призначені для модернізації мікропроцесорної системи або комплекс. Модернізація здійснюється заміною початкової моделі МП на нові моделі, так звані OverDrive-процесори, які мають вищі техніко-економічні показники. Такими моделями для *i*486 є Intel DX2 OverDrive, Intel DX4 OverDrive, Pentium OverDrive 63 та 83 МГц.

Для модернізації мікропроцесорної системи на системній платі, крім уже встановленого процесора, у додатковий роз'єднувач (сокет), позначений як OverDrive, встановлюється OverDrive-процесор у корпусі PGA-169. OverDrive-процесор спеціальним вихідним сигналом від'єднує основний процесор, що залишається на платі.

Режим системного керування 32-розрядних мікропроцесорів. Деякі модифікації процесорів 386 і 486, крім перелічених режимів (реального, захищеного і віртуального *V*86), мають режим системного керування *SMM (System Management Mode)*, який призначено для керування енергоспоживанням або виконання програм, повністю ізольованих як від прикладного програмного забезпечення, так і від операційної системи. Перемикання в режим *SMM* здійснюється або апаратно — подачею нульового потенціалу на контакт $\overline{SMI}$ мікросхеми процесора, або, в деяких моделях, програмно — за прийняттям повідомлення по шині *APIC*. Під час переходу в режим *SMM* МП виставляє сигнал підтвердження на контакті $\overline{SMIACT}$. Після цього процесор зберігає свій стан — уміст майже всіх регістрів — у спеціальній області пам'яті *SMRAM*. Якщо режим *SMM* використовується для вимкнення живлення процесора з можливістю швидкого ввімкнення, пам'ять *SMRAM* має бути енергонезалежною. У цій області пам'яті знаходиться підпрограма обробки переривання *SMI*. Доступ до пам'яті дозволений лише за наявності сигналу $\overline{SMIACT}$. Повернення з режиму *SMM* здійснюється як програмно, так і за перериванням.

3.4. Особливості архітектури мікропроцесорів Pentium

Мікропроцесор Pentium — високопродуктивний 32-розрядний процесор з внутрішньою 64-розрядною шиною даних. Процесор є продовженням розробок процесорів *i*80 × 86 і програмно-сумісний з ними, але має певні особливості. У МП Pentium уперше застосовано 0,8 мкм BiCMOS-технологію, яка поєднує переваги двох технологій — швидкодію біполярної і мале енергоспоживання CMOS. Використання субмікронної технології дало змогу збільшити кількість транзисторів до 3,1 млн. Для порівняння: процесор 8086 містить 29 тис. транзисторів, а найближчий до Pentium процесор *i*486 — 1,2 млн транзисторів. Збільшення кількості транзисторів (більше ніж удвічі) дало змогу розмістити в одній мікросхемі компоненти, що раніше розташовувалися в інших мікросхемах. Це зменшило тривалість доступу і збільшило продуктивність процесора. Висока тактова частота, суперскалярна архітектура, роздільна кеш-пам'ять для програм і даних та інші вдосконалення сприяли більшій продуктивності та сумісності з програмним забезпеченням, розробленим для мікропроцесорів фірми Intel. Мікропроцесор Pentium дає змогу використовувати такі операційні системи, як UNIX, Windows NT, OS/2, Solaris і NEXTstep. Розглянемо особливості архітектури.

Структурна схема і характеристики. Узагальнена структурна схема МП Pentium (рис. 3.16) містить:

- ШІ — 64-розрядний шинний інтерфейс;
- два 32-розрядних цілочислових АЛП;
- кеш-пам'ять команд;
- кеш-пам'ять даних;
- РЗП;
- буфери вибірки з випередженням (БВВ);
- блок передбачення адреси переходу (БПАП);
- блок конвеєрних обчислень з плаваючою комою (БКОПК).

Шинний інтерфейс призначений для сполучення внутрішньої шини процесора із зовнішньою шиною.

Розширена 64-разрядна шина даних. Завдяки цьому МП Pentium підтримує кілька типів циклів шини, включаючи і пакетний режим, за якого частина даних з 256 біт передається у кеш-пам'ять даних за один цикл. Це істотно підвищує швидкість передавання порівняно з процесором *i*486 *DX*. Наприклад, МП Pentium з частотою шини 66 МГц має швидкість передавання 528 Мбайт/с, МП *i*486 *DX* з частотою шини 50 МГц — 160 Мбайт/с. Розширена шина даних забезпечує конвеєри-

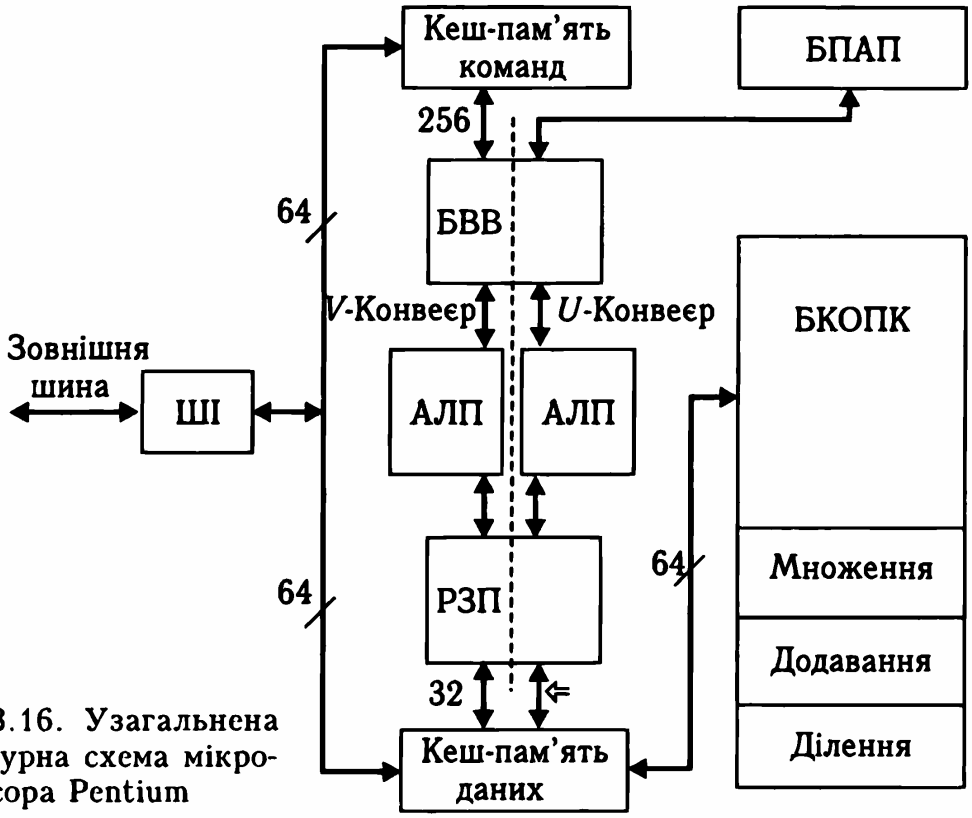


Рис. 3.16. Узагальнена структурна схема мікропроцесора Pentium

зацію циклів шини, що збільшує пропускну здатність шини і дає змогу другому циклу розпочинатися раніше, ніж завершився перший.

Суперскалярна архітектура. Термін «суперскалярна» означає мікропроцесорну архітектуру, що містить більше, ніж один обчислювальний блок. Процесор Pentium має два конвеєри, які можуть виконувати дві команди одночасно — *U*-конвеєр з повним набором і *V*-конвеєр з обмеженим набором команд. На рис. 3.16 конвеєри спрощено подано двома цілочисловими АЛП, РЗП і ББВ. Як і у випадку єдиного конвеєра процесора *i*486, подвійний конвеєр процесора Pentium виконує цілочислові команди у п'ять етапів (рис. 3.17):

- вибірка з випередженням команди з пам'яті (передвибірка) *PF* (*PreFetch*);
- декодування команди (стадія 1) *D*1;
- декодування команди (стадія 2) *D*2;
- виконання команди *EX*;
- запам'ятовування результату в буфері відкладеного запису *WB* (див. п. 3.3).

Перший етап виконується блоком ББВ, який має чотири 32-розрядних буфери. Дві незалежні пари буферів вибірки працюють сумісно з БПАП, який передбачає, буде перехід чи ні. Якщо перехід не передбачається, продовжується вибірка, якщо передбачається — то дозволяється робота іншого бу-

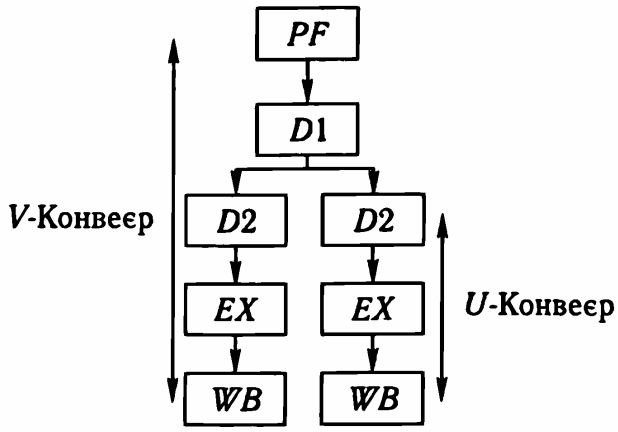


Рис. 3.17. Етапи виконання цілочислової команди у процесорі Pentium

фера, і він починає передвибірку з точки переходу. Якщо передбачений перехід не здійснився, конвеєри команд очищаються, і передвибірка починається знову. На другій стадії декодування команди формуються адреси операндів пам'яті.

Кожний конвеєр має свій 64-розрядний буфер відкладеного запису, які можуть заповнюватися за один такт, наприклад, за одночасних кеш-промахів запису на обох конвеєрах. Ніякі запити на читання не порушують порядку запитів на запис, які вже знаходяться у буфері. Pentium підтримує суворий порядок запису.

Високопродуктивний математичний співпроцесор БКОПК містить 8-тактовий конвеєр та апаратні засоби реалізації арифметичних операцій — множення, додавання, ділення. Більша частина команд операцій з плаваючою комою можуть виконуватися в одному цілочисловому конвеєрі, після чого вони надходять у конвеєр обчислень з плаваючою комою. Продуктивність вбудованого арифметичного співпроцесора Pentium переважає продуктивність математичного співпроцесора FPU-486 (*Floating-Point Unit*) у 2—10 разів.

Використання подвійного конвеєра дає змогу кільком командам знаходитися у різних стадіях виконання і додатково збільшувати продуктивність МП повним заповненням конвеєрів командами. У процесорі Pentium використовується апаратне виконання команд, що також підвищує продуктивність процесора.

Роздільні кеш-пам'яті для команд і даних. Мікропроцесор Pentium має розділені кеш-пам'яті команд і даних. Це дає змогу уникнути конфліктів між процесом вибірки для однієї команди і доступом до даних для іншої, які можуть виникати, наприклад, у процесорі *i*486. Під час реалізації роздільних кеш-пам'яті для команд і даних обидві команди можуть виконуватися одночасно. Ємність кеш-пам'яті команд та кеш-пам'яті даних у процесорі Pentium однакова і становить 8 Кбайт. Кеш-пам'ять команд і даних виконано за схемою двоканальної асоціативної кеш-пам'яті (див. п. 3.5), причому кеш-пам'ять даних має два інтерфейси (по одному для

кожного конвеєра), що дає змогу забезпечувати даними дві окремі команди впродовж одного машинного циклу.

Кеш-пам'ять даних працює з відкладеним (до звільнення зовнішньої шини) записом і налагоджується у режим наскрізного або зворотного запису. В останньому випадку дані зчитуються з кеш-пам'яті, а після цього записуються в основну пам'ять. Такий спосіб використання кеш-пам'яті сприяє збільшенню продуктивності порівняно з кеш-пам'яттю з безпосереднім записом, за якого процесор записує дані одночасно в кеш-пам'ять та основну пам'ять. Кеш-пам'ять даних підтримує протокол *MESI*, який забезпечує роботу з урахуванням можливості звернення інших процесорів до кеш-пам'яті даних. Назва протоколу *MESI* складається з назв станів рядка кеш-пам'яті: *M* (*Modified*), *E* (*Exclusive*), *S* (*Shared*), *I* (*Invalid*). Стани рядка кеш-пам'яті визначаються так:

M-стан — рядок, наявний лише у кеш-пам'яті процесора, що розглядається. Рядок модифікований, тобто відрізняється від вмісту основної пам'яті. Запис у нього можливий без генерації зовнішнього (щодо локальної шини) циклу звернення;

E-стан — рядок, наявний лише у кеш-пам'яті процесора, що розглядається, але він не модифікований. Запис у рядок можливий без генерації зовнішнього циклу звернення. Під час запису в рядок він переходить у *M*-стан;

S-стан — рядок, наявний у кеш-пам'яті процесора, що розглядається, і потенційно може бути наявним у кеш-пам'яті інших процесорів. Його читання можливе без генерації зовнішнього циклу, а запис має супроводжуватися наскрізним записом в основну пам'ять, що зумовлює анулювання відповідних рядків у кеш-пам'яті інших процесорів;

I-стан — рядок відсутній у кеш-пам'яті, його читання з основної пам'яті може привести до генерації циклу заповнення рядка кеш-пам'яті. Запис у рядок кеш-пам'яті наскрізний з використанням зовнішньої шини.

Підтримування мультипроцесорного режиму роботи. Архітектура МП Pentium дає змогу працювати двом і більше Pentium-процесорам у мультипроцесорних системах. Реалізовано інтерфейс побудови двопроцесорних систем із симетричною архітектурою (починаючи з другого покоління МП Pentium).

Засоби задання розміру сторінки пам'яті. Pentium-процесор має опцію (спеціальний біт керування) для вибору розміру сторінок пам'яті — традиційну (4 Кбайт) і розширену (4 Мбайт). Збільшення розміру сторінки доцільне під час використання громіздких графічних додатків.

Засоби виявлення помилок і тестування за допомогою функціональної надмірності. З метою підвищення надійності у процесорі Pentium передбачено внутрішнє виявлення помилок внутрішніх пристроїв (внутрішній контроль паритету) та зовнішнього шинного інтерфейсу, контроль паритету шини адреси та тестування за допомогою функціональної надмірності. *Внутрішнє визначення помилок* полягає у доповненні кодів команд і даних бітом парності, що дає змогу визначати помилки непомітно як для системи, так і для користувача. *Тестування за допомогою функціональної надмірності* використовується у програмних додатках, особливо критичних щодо достовірності результатів. Тестування за допомогою функціональної надмірності ґрунтується на роботі двох Pentium-процесорів у конфігурації *головний-контрольований (master/checker)*. У такій конфігурації основний процесор працює у звичайному однопроцесорному режимі. Контрольований процесор виконує ті самі операції, але не керує шиною, і порівнює вихідні сигнали основного процесора з тими сигналами, які він генерує сам. У разі розбіжності отриманих результатів формується сигнал помилки, який може оброблятися системою як переривання. Такий спосіб дає змогу виявляти понад 99 % помилок. Крім того, засоби тестування передбачають можливість виконання вбудованого теста *BIST* (*Built In Self Test*), що забезпечує виявлення помилок мнемокодів, програмованих логічних матриць, тестування кеш-пам'яті команд і даних, адресних буферів і ПЗП. У цілому самотестування охоплює понад 70 % вузлів процесора. Усі процесори мають стандартний тестовий порт *IEEE* 1149.1 для самотестування за допомогою стандартного інтерфейсу *JTAG*.

Особливостями процесорів Pentium є:

- введення кількох нових команд, у тому числі розпізнавання моделі процесора;
- введення засобів керування енергоспоживанням;
- застосування конвеєрної адресації шинних циклів;
- скорочена тривалість (кількість тактів) виконання команд;
- трасування команд і моніторинг продуктивності;
- розширення можливостей віртуального режиму — впровадження віртуалізації прапорця переривань.

Реалізовано нові додаткові засоби налагодження:

зондовий режим (*Probe Mode*), що забезпечує доступ до внутрішніх регістрів, ПВВ і системної пам'яті процесора. Цей режим дає змогу перевіряти і змінювати стан процесора за допомогою засобів налагодження програм з можливостями, подібними до можливостей внутрішньосхемних емуляторів;

розширені налагодження *DE* (*Debug Extensions*), які дають змогу ставити контрольні точки за адресами введення-виведення;

внутрішні лічильники, які використовують для поточного контролю продуктивності та обліку кількості подій;

покрокове виконання за допомогою команди *CPUID*.

Розширення архітектури. Додатково до базової архітектури 32-розрядних процесорів (див. п. 3.2) Pentium має набір регістрів, специфічних для моделі MSR (*Model Specific Registers*). Склад регістрів *MSR* може бути неоднаковим у різних моделях МП (Pentium і Pentium Pro), яке призводить до їхньої можливої несумісності. Програмне забезпечення, що використовує регістри *MSR*, має використовувати відомості про процесор, отримані за допомогою команди *CPUID*.

До складу регістрів *MSR* входять:

- тестові регістри *TR*1—*TR*12 (див. п. 3.2);
- засоби моніторингу продуктивності;
- регістри-фіксатори адреси і даних циклу, який зумовив спрацювання контролю машинної помилки.

Тестові регістри дають змогу керувати більшістю функціональних вузлів процесора, забезпечуючи можливість докладного тестування їх працездатності. За допомогою бітів регістра *TR*12 можна заборонити нові архітектурні властивості (передбачення і трасування розгалужень, паралельне виконання команд), а також роботу кеш-пам'яті.

Засоби моніторингу продуктивності сприяють оптимізації апаратного та програмного забезпечення завдяки виявленню у програмному коді потенційно «вузьких місць». Розробник може спостерігати та підраховувати такти внутрішніх процесорних подій, які впливають на продуктивність операцій читання і запису, вдалі та невдалі звернення до кеш-пам'яті, переривання, використання шини. Це дає змогу оцінювати ефективність програмного коду і досконало налагоджувати програмні додатки або системи для отримання максимальної продуктивності. Засобами моніторингу продуктивності є таймер реального часу і лічильники подій. Таймер *TSC* (*Timer Stamp Counter*) виконано на базі 64-розрядного лічильника, вміст якого інкрементується з кожним тактом роботи ядра процесора. Для читання його вмісту існує команда *RDTSC*; 40-розрядні лічильники подій *CTR*0, *CTR*1 програмуються на підраховування подій різних класів, пов'язаних із шинними операціями, виконанням команд, роботою конвеєрів, кеш-пам'яті, контролем точок зупинки тощо. Шестибітові поля типів подій дають змогу кожному з лічильників незалежно підраховувати події з великого списку. Стан лічильників можна

попередньо встановлювати і зчитувати програмно. Крім того, існують зовнішні лінії *PM*1—*PM*0, які програмуються на зазначення фактів спрацьовування або переповнення відповідних лічильників. Оскільки ці сигнали можуть змінювати своє значення з частотою, що не перевищує частоту системної шини, через внутрішнє множення частоти кожна поява цих сигналів може означати і кілька (до значення коефіцієнта множення) фактів спрацювання лічильників.

Регістри-фіксатори адреси і даних циклу. Назва регістрів-фіксаторів адреси і даних циклу, що зумовив спрацювання контролю машинної помилки, вказує на їх можливу несумісність для різних класів (Pentium та Pentium Pro) або навіть для різних моделей процесорів. Програма, що їх використовує, має звернутися до відомостей про процесор за командою *CPUID*.

Процесори Pentium мають можливість зменшувати енергоспоживання у неробочому режимі. За сигналом *STOPCLK#* процесор вивантажує буфери відкладеного запису і входить у режим *Stop Grant*, в якому припиняється тактування більшості вузлів процесора, що сприяє зниженню енергоспоживання приблизно у 10 разів. У цьому стані МП припиняє виконання команд і не обслуговує переривання, однак продовжує спостереження за шиною даних. З цього стану процесор виходить після закінчення сигналу *STOPCLK#*. Керування сигналом *STOPCLK#* разом з використанням режиму *SMM* реалізує механізм *розширеного керування живленням APM (Advanced Power Management)*. Для уповільнення процесора з пропорційним зниженням споживаної потужності сигнал *STOPCLK#* має бути періодично імпульсним. Скважність імпульсів визначає коефіцієнт простою процесора і його продуктивність.

У стан зниженого споживання *Auto HALT PowerDown* процесор переходить під час виконання команди *HALT*. У цьому стані процесор реагує на всі переривання і також продовжує спостереження за шиною. У режимі припинення зовнішньої синхронізації процесор споживає мінімальну потужність, але не виконує ніяких функцій. Наступне подавання сигналу синхронізації супроводжується сигналом апаратного скидання *RESET*.

Основні технічні характеристики процесорів Pentium наведено у табл. 3.9.

Процесори Pentium P5 з тактовою частотою 60 та 66 МГц мають напругу живлення 5 В і потребують примусового охолодження. Ці процесори випускаються у корпусах PGA-273 (матриця 21 × 21).

Таблиця 3.9. **Основні технічні характеристики процесорів Pentium**

Процесор	Розрядність			Ємність кеш-пам'яті, Кбайт	Передбачення переходів	Підтримка *MMX*	Частота, МГц	
	регістрів	шини даних	шини адреси				шини	процесорного ядра
Pentium (1-ше покоління)	32	64	32	*L*1: 2 × 8	+	–	60 66	60 66
Pentium (2-ге покоління)	32	64	32	*L*1: 2 × 8	+	–	 50 60 66	75 90 100 120, 133, 150, 166, 180, 200
Pentium OverDrive	32	64	32	*L*1: 2 × 8	+	–	60 66	120 133
Pentium MMX	32	64	32	*L*1: 2 × 16	+	+	66	166, 200, 233
Pentium Pro	32	64	36	*L*1: 2 × 8 *L*2: 256 (512)	+	–	50 60 66	150 166 180, 200
Pentium II OverDrive	32	64	36	*L*1: 2 × 16 *L*2: 512	+	+	66	333
Pentium II	32 64*	64	36	*L*1: 2 × 16 *L*2: 512	+	+	66 100	233 266 300 350, 400, 450

Продовж. табл. 3.9

Процесор	Розрядність			Ємність кеш-пам'яті, Кбайт	Передбачення переходів	Підтримка *MMX*	Частота, МГц	
	регістрів	шини даних	шини адресси				шини	процесорного ядра
Pentium III	32 64*	64	36	*L*1 2 × 16, *L*2 256 (512)	+	+	100 (133)	450 500 533, 550, 600, 650, 667, 700, 733, 750, 800, 850, 866, 933, 1000, 1130
Pentium II Xeon	32 64*	64	36	*L*1: 2 × 16, L2: 512 (1024)	+	+	66 100	233 266 300 350, 400
Pentium III Xeon	32 64*	64	36	*L*1: 2 × 16, *L*2: 256 (1 Мбайт) (2 Мбайт)	+	+	100 (133)	550 600 650 667, 700, 733, 750, 800, 866, 933, 1000
Celeron	32 64*	64	36	*L*1: 2 × 16, *L*2: 128**	+	+	66	233 266, 300, 350, 400, 450, 500, 533, 566, 600, 633, 667, 700

AMD K5	32	64	32	*L*1: 16 /8	+	–	50 60 66	75 90 100, 120, 133, 166
AMD K6 MMX	32	64	36	*L*1: 2 × 32, *L*2: 256	+	+ 3*DNow!*	66	166 200, 233, 266, 300
AMD K6-2	32	64	36	*L*1: 2 × 32	+	+ 3*DNow!*	66 100	200 300, 333, 350, 366, 400, 450, 475, 500, 550
AMD K6-2+	32	64	36	*L*1: 2 × 32, *L*2: 128	+	+ 3*DNow!*	66 100	450 – 550
AMD K6-III	32	64	36	*L*1: 2 × 32, *L*2: 256	+	+ 3*DNow!*	66 100	350 – 475
Cyrix 6 × 86MX	32 64*	64	36	*L*1: 16	+	+	60 66 75 83	16 200 233 266
Cyrix 6 × 86MIIxxxGP	32 64*	64	36	*L*1: 16	+	+	66 75 83	300 – 433
VIA Cyrix III	32 64*	64	36	*L*1: 2 × 64	+	+ 3*DNow!*	66 100 133	500 – 600

Примітки: * Розрядність 64 мають лише регістри *MMX*.
** У процесорів Celeron 266 і 300 кеш-пам'ять рівня *L*2 відсутня:
*L*1 – перший рівень (роздільна кеш-пам'ять команд і даних);
*L*2 – другий рівень (спільна кеш-пам'ять команд і даних).

Процесори Pentium P54 з тактовою частотою 75...200 МГц мають напругу живлення 3,3 В, що знижує потужність тепловиділення. Процесори містять розширені засоби керування енергоспоживанням *SMM* (*System Management Mode*). Засоби внутрішнього множення частоти дають змогу реалізувати роботи зовнішньої системної шини на частотах 50, 60 і 66,6 МГц, а ядра процесора — на частотах 75, 90, 100, 120, 133, 150, 166, 200 МГц.

Процесори можуть працювати у двопроцесорних системах, підтримуючи режим *SMP* (*Symmetric Multi-Processing*) або тестування за допомогою функціональної надмірності *FRC* (*Functional Redundancy Checking*). У режимі *SMP* кожний процесор виконує свою задачу, але обидва використовують спільні ресурси комп'ютера, включаючи пам'ять і зовнішні пристрої. У кожний момент часу шиною може керувати лише один процесор з двох, але вони можуть мінятися ролями. Для реалізації двопроцесорного режиму введено *розширений програмний контролер переривань APIC* (*Advanced Programmable Interruption Controller*). Цей контролер має зовнішні сигнали локальних переривань *LINT* [1 : 0] та трипровідну інтерфейсну шину (*PICD* [1 : 0] і *PICCLK*), за допомогою якої обидва процесори з'єднані з контролером APIC системної плати. Запити локальних переривань обслуговуються лише тим процесором, на виводи якого (*LINT*0, *LINT*1) надходять їхні сигнали. Режим оброблення апаратних переривань *APIC* дозволяється сигналом *APICEN*; надалі він може бути заборонений програмно.

Pentium OverDrive-процесори (див. п. 3.3) з частотами 120 і 133 МГц — варіанти процесорів Pentium другого покоління зі зниженим енергоспоживанням і подвоєнням частоти призначені для заміни процесорів Pentium першого покоління. Ці процесори дорожчі від звичайних моделей Pentium з частотою 120 або 133 МГц. Їхнє застосування доцільне лише тоді, коли з будь-яких причин неможливо замінити системну плату, а продуктивності МП Pentium з частотою 60 або 66 МГц недостатньо. Pentium OverDrive з частотою 125, 150 та 166 МГц призначені для заміни процесорів першого покоління із частотами 75, 90 та 100 МГц.

Процесор Pentium Pro (P6). У цьому процесорі застосоване *динамічне виконання команд*, тобто комбінація засобів передбачення множинних відгалужень, аналізу проходження даних і віртуального виконання, за якого в процесорі команди можуть виконуватися не в такому порядку, як це передбачено програмним кодом. При цьому команди, що не залежать від результатів попередніх операцій, виконуються у змішено-

му порядку, але послідовність вивантаження результатів у пам'ять та порти відповідають початковому програмному коду. Можливість виконання команд з випередженням (спекулятивне виконання), перевпорядкування команд у разі, якщо команду в одному конвеєрі буде виконано швидше, ніж попередню у другому конвеєрі, і передбачення переходів за динамічного виконання підвищує продуктивність обчислень.

Процесор підтримує режим тестування за допомогою функціональної надмірності *FRC*. Архітектура процесора сприяє об'єднанню у симетричну мультипроцесорну систему до чотирьох процесорів на одній шині. Сумарну пропускну здатність підвищено за рахунок подвійної незалежної шини. Одна шина (системна) виконує функцію взаємодії ядра процесора з основною пам'яттю та інтерфейсними пристроями, друга — виключно функцію обміну з вторинною кеш-пам'яттю. Перша шина працює на тактовій частоті процесора, друга — на частоті системи. Таке розділення шин дає змогу втричі прискорити обмін процесора з пам'яттю. Завдяки цьому відпадає потреба в окремій зовнішній кеш-пам'яті. Мікропроцесор містить роздільні кеш-пам'яті першого рівня (*L*1) для даних і команд ємністю 8 Кбайт кожна та об'єднану кеш-пам'ять другого рівня (*L*2). Кеш-пам'ять даних першого рівня підтримує одну операцію завантаження та одну операцію запису за такт. Інтерфейс кеш-пам'яті другого рівня працює з тактовою частотою процесора і може передавати 64 біт за один такт. Ємність кеш-пам'яті другого рівня становить 256 Кбайт (для технології 0,6 мкм) та 512 Кбайт (для технології 0,35 мкм).

Процесори Pentium MMX (P55C) — процесори, зорієнтовані на мультимедійне, 2*D*- і 3*D*-графічне та комунікаційне застосування. В архітектуру Pentium MMX додатково введено:

- вісім 64-розрядних *MMX*-регістрів, які розташовані у 64 молодших розрядах стеку 80-розрядних регістрів блока обчислень з плаваючою комою;
- чотири нових типи даних (упаковані байти — 8 байт у 64-розрядному пакеті; упаковані слова — чотири 16-розрядних слова у 64-розрядному пакеті; упаковані подвійні слова — два 32-розрядних подвійних слова у 64-розрядному пакеті; 64-розрядні слова);
- 57 додаткових команд, які поділяють на такі групи: обміну даними між регістрами *MMX* та цілочисловими регістрами або пам'яттю, арифметичні (додавання та віднімання у різних режимах, множення, комбінації множення та додавання), порівняння, перетворення форматів, логічні, зсуву (логічного та арифметичного), очищення регістрів *MMX*;

• подвоєні ємності кеш-пам'яті команд і кеш-пам'яті даних (16 Кбайт);

• поліпшену логіку передбачення переходів;

• розширену конвеєризацію;

• глибшу буферизацію пам'яті (подвоєний розмір буфера відкладеного запису).

Арифметичні та логічні операції у процесорі Pentium MMX виконуються паралельно над кожним байтом слова або подвійного слова, що міститься у 64-розрядному *MMX*-регістрі. Так реалізується *SIMD*-модель оброблення (*Single Instruction — Multiple Data*). У *SIMD*-командах обробляється одночасно 64-розрядне слово, яке залежно від *MMX*-команди трактується як вісім однобайтових операндів, чотири двобайтових, два чотирибайтових або один восьмибайтовий операнд. *SIMD*-Оброблення істотно прискорює виконання мультимедійних алгоритмів, для яких є характерним здійснення ідентичних операцій над великими масивами однотипних даних, наприклад, 16-розрядними відліками оцифрованого звука, 8-розрядними кодами кольору пікселя та ін. Використання *MMX*-команд дає змогу підвищити швидкість виконання мультимедійних операцій на 60 % порівняно з процесором Pentium першого покоління за однакових тактових частот. В інших командах забезпечується сумісність з процесором Pentium. Команди *MMX* не впливають на прапорці і не зумовлюють переривань та винятків і доступні для будь-якого режиму роботи процесора.

Поліпшена логіка передбачення переходів забезпечується збільшеною кількістю буферів вибірки команд з випередженням. Процесор Pentium MMX має чотири 16-розрядних БВВ.

Розширена конвеєризація забезпечується збільшенням етапів виконання цілочислових програм у конвеєрі до шести за рахунок додавання етапу вибірки (*F*) між етапами передвибірки (*PF*) та декодування команди (*D*1). На етапі вибірки виконується декодування довжини команди, яке у попередніх моделях процесора Pentium виконується на етапі *D*1.

Процесор Pentium MMX має подвоєний порівняно з процесором Pentium розмір обох частин кеш-пам'яті *L*1 і *L*2. Однак у процесорі Pentium MMX немає режиму тестування за допомогою функціональної надмірності *FRC*. У двопроцесорних системах підтримується лише режим *SMP*.

Pentium® OverDrive® MMX процесори — варіант процесорів Pentium MMX з тактовими частотами 150, 166, 180 і 200 МГц для заміни звичайних (без розширення *MMX*)

процесорів Pentium з частотою 75—200 МГц. Вони вирізняються фіксованим коефіцієнтом множення частоти (3) і відсутністю двопроцесорних режимів.

Процесори Pentium для мобільних застосувань (блокнотних персональних комп'ютерів) мають знижене енергоспоживання внаслідок зниження напруги живлення ядра процесора. Вони характеризуються вищою допустимою температурою, що дає змогу використовувати їх у жорсткіших умовах експлуатації. Слід зазначити, що ці процесори двопроцесорний режим не підтримують і не мають розширеного програмованого контролера переривань *APIC* та відповідних зовнішніх виводів.

Процесор Pentium II має 36-розрядну шину адреси, що дає змогу адресувати фізичну пам'ять ємністю 64 Гбайт. Частота ядра процесора становить 233, 266, 300 та 450 МГц, частота шини — 66 або 100 МГц. Висока продуктивність процесора досягається завдяки застосуванню у ньому динамічного виконання команд, розширення *MMX* та подвійної незалежної шини.

Кеш-пам'ять першого рівня (*L*1) збільшено до 32 Кбайт (16 Кбайт — кеш-пам'ять команд, 16 Кбайт — кеш-пам'ять даних). Кеш-пам'ять другого рівня (*L*2) розташовано на одній платі з процесором.

Процесор Pentium III виконано за технологією *SSE* (*Streaming SIMD Extensions*). У процесора Pentium III реалізовано 70 нових *SIMD*-команд, що оперують з 128-розрядними регістрами *XMM*0—*XMM*7. Отже, виконуючи операцію над двома регістрами, *SSE* фактично оперує чотирма парами чисел. Завдяки цьому процесор може виконувати до чотирьох операцій одночасно:

- тривимірна графіка і моделювання, які використовують обчислення у форматі з плаваючою комою;
- обробка сигналів і моделювання процесів із широким діапазоном зміни параметрів;
- генерація тривимірних зображень у програмах реального часу, що не використовують цілочисловий код;
- алгоритми кодування і декодування відеосигналів з блочною обробкою;
- числові алгоритми цифрової фільтрації, що працюють з потоками даних.

Крім розглянутих процесорів фірми Intel є процесори, аналогічні за продуктивністю процесорам Pentium, які випускають інші фірми — AMD (AMD K5 PR 75/90/100, AMD K6), CYRIX (C×86 (M1), CYRIX 6×86, P120+, P133+, P150+, P166+, P200+, CYRIX 6×86MX, CYRIX MediaGX),

Hewlett Packard (Merced (P7), PA RISC, PA-8000), DEC (Alpha 21068, Alpha 21164, Alpha 21264).

Процесори AMD K5 PR75/90/100/120/133/166 виконані за гібридною *CISC/RISC*-архітектурою і мають ускладнений конвеєр з п'ятьма блоками оброблення даних, що функціонують паралельно. На відміну від процесорів Pentium у цих процесорах можуть одночасно виконуватися команди з плаваючою комою, завантаження (зберігання) та переходу. Великий набір і можливість динамічного перейменування регістрів і наявність блока завантаження (зберігання) дають змогу виконувати дві операції за один цикл вибірки з пам'яті. У процесорі використовується передбачення розгалужень та зміна порядку виконання команд. Кеш-пам'ять команд має ємність 16 Кбайт, а кеш-пам'ять даних — 8 Кбайт.

Процесори AMD-K6 MMX мають роздільну кеш-пам'ять даних і команд першого рівня ємністю по 32 Кбайт. Кеш-пам'ять даних є двопортовою і підтримує зворотний запис. Кеш-пам'ять команд має додаткову область для попередньо декодованих команд. Кожна команда має біти передекодування, які вказують на зміщення початку наступної команди у кеш-пам'яті. Внутрішньої вторинної кеш-пам'яті немає. Процесор підтримує логіку передбачення розгалужень, використовуючи таблицю історії розгалужень з 8192 елементів; кеш-пам'ять адрес переходу та стек повернення, які забезпечують імовірність правильного передбачення переходу понад 95 %.

На відміну від процесорів Intel P54 і Intel P55, процесор AMD-K6 MMX вбудованих засобів підтримки мультипроцесорних систем не має, включаючи APIC. У ньому немає сигналу перевірки шинних операцій (BUSCHK) і зондового режиму, а також не виводяться сигнали точок зупинки та моніторингу продуктивності. У процесорі використовується спекулятивне виконання зі зміною послідовності команд, попереднє посилання даних, перейменування регістрів.

Процесори AMD-K6-2 з технологією 3DNow! виконано за технологією 0,25 мкм. Вони характеризуються швидкодіючою вбудованою кеш-пам'яттю другого рівня ємністю 256 Кбайт, кеш-пам'яттю першого рівня ємністю 64 Кбайт. Процесор має ефективну *RISC* архітектуру і поліпшений блок обчислень з плаваючою комою. Частота ядра процесора має значення від 300 до 400 МГц. Частота шини — 100 МГц.

Процесори Cyrix 6 × 86 (M1) мають можливість динамічного перейменування регістрів, зміни порядку виконання команд, спекулятивне виконання, передбачення переходів.

Процесори містять дві кеш-пам'яті: уніфіковану первинну кеш-пам'ять ємністю 16 Кбайт, що використовується як для команд, так і для даних, та додаткову 256-байтову кеш-пам'ять команд. Виділена кеш-пам'ять команд дає змогу уникати конфліктів у разі звернення до даних і команд у спільній кеш-пам'яті. Процесор здатний одночасно виконувати цілочислові команди та команди з плаваючою комою. Етапи циклу виконання команди такі:

- вибірка команди *F*;
- декодування команди (стадія 1) *D*1;
- декодування команди (стадія 2) *D*2;
- обчислення адреси (стадія 1) *AC*1;
- обчислення адреси (стадія 2) *AC*2;
- виконання *EX*;
- запис результату *WB*.

Етапи декодування та обчислення адреси конвеєризовано, причому забезпечено можливість перевпорядкування команд на етапах *EX* і *WB*. Існує версія низьковольтних процесорів для мобільних застосувань.

Процесори Cyrix 6 × 86MX — удосконалений варіант процесора M1, в який додатково введено можливість виконання набору з 57 мультимедійних команд, сумісних з *MMX*-розширенням, та підвищено тактову частоту. Процесор містить дві кеш-пам'яті: чотиривходову асоціативну кеш-пам'ять ємністю 64 Кбайт зі зворотним записом та високошвидкісну кеш-пам'ять команд ємністю 256 байт. Для мобільних застосувань передбачено ефективну систему керування енергоспоживанням.

3.5. Особливості архітектури 64-розрядних мікропроцесорів

У 1997 р. фірми Intel і Hewlett-Packard розробили нову мікропроцесорну архітектуру *EPIC (Explicitly Parallel Instruction Computing* — явного паралельного обчислення інструкцій), яку було покладено в основу 64-розрядних мікропроцесорів IA-64, McKinley, Itanium, Itanium 2.

Особливостями архітектури *EPIC* є:

велика кількість регістрів загального призначення. Так, кількість регістрів МП IA-64 містить 128 64-розрядних регістрів для операцій з цілими числами і 12880 — з дробовими;

пошук залежностей між командами, причому пошук виконує не процесор, а компілятор. Команди МП IA-64

групуються компілятором у «зв'язку» завдовжки 128 розрядів. Зв'язка містить три команди і шаблон, в якому зазначені залежності між командами (тобто визначається, чи можна з командою к1 виконати паралельно команду к2 або команда к2 має виконатися лише після команди к1), а також між іншими зв'язками (чи можна з командою к3 зі зв'язки с1 виконати паралельно команду к4 зі зв'язки с2);

масштабованість архітектури, тобто пристосування набору команд до великої кількості функціональних пристроїв. Наприклад, одна зв'язка з трьох команд відповідає наборові з трьох функціональних пристроїв процесора. Процесори IA-64 можуть мати різну кількість таких функціональних пристроїв, залишаючись при цьому сумісними за кодом. Завдяки тому, що в шаблоні зазначена залежність і між зв'язками, процесору з N однаковими блоками з трьох функціональних пристроїв відповідатиме командне слово з N × 3 команд (N зв'язок);

предикація *(Predication). Предикацією* називають спосіб обробки умовних розгалужень. Команди з різних гілок умовного розгалуження позначаються предикатними полями (полями умов) і виконуються паралельно, але їхні результати не записуються, доки значення предикатних регістрів не визначені. Якщо наприкінці циклу визначається умова розгалуження, предикатний регістр, який відповідає «правильній» гілці, встановлюється у стан логічної одиниці, а другий — у стан логічного нуля. Перед записом результатів процесор перевіряє предикатне поле і записує результати лише тих команд, предикатне поле яких містить одиницю;

завантаження за припущенням *(Speculative loading).* Цей механізм призначений знизити простої процесора, пов'язані з чеканням виконання команд завантаження з відносно повільної основної пам'яті. Компілятор переміщує команди завантаження даних з пам'яті так, щоб вони виконувалися якомога раніше. Отже, якщо дані з пам'яті знадобляться будь-якій команді, процесор не простоюватиме.

Процесор Itanium 2, виконаний за 0,18 мкм технологією, здатний виконувати шість команд за один машинний цикл. У сукупності з підвищенням тактової частоти та пропускної спроможності системної шини (6,4 Гб/с, частота шини — 400 МГц, розрядність шини — 128), цей чинник забезпечує в 1,5—2 рази більшу продуктивність, ніж у процесорі Itanium. Процесор має велику ємність кеш-пам'яті третього рівня, розміщеної на кристалі (до 3 Мбайт працює на частоті ядра).

У майбутньому на ринку з'являться процесори, зроблені за технологією 0,12 мкм Deerfield та призначені для використання в двопроцесорних системах і Madison, орієнтовані на багатопроцесорні системи. Процесор Montecito буде виготовлятися з використанням технології 90 нм.

64-Розрядні МП сімейства Hammer, розроблені фірмою AMD, ґрунтуються на архітектурі *x*86-64, яка є розширенням архітектури 32-розрядних процесорів *x*86-32 (рис. 3.18).

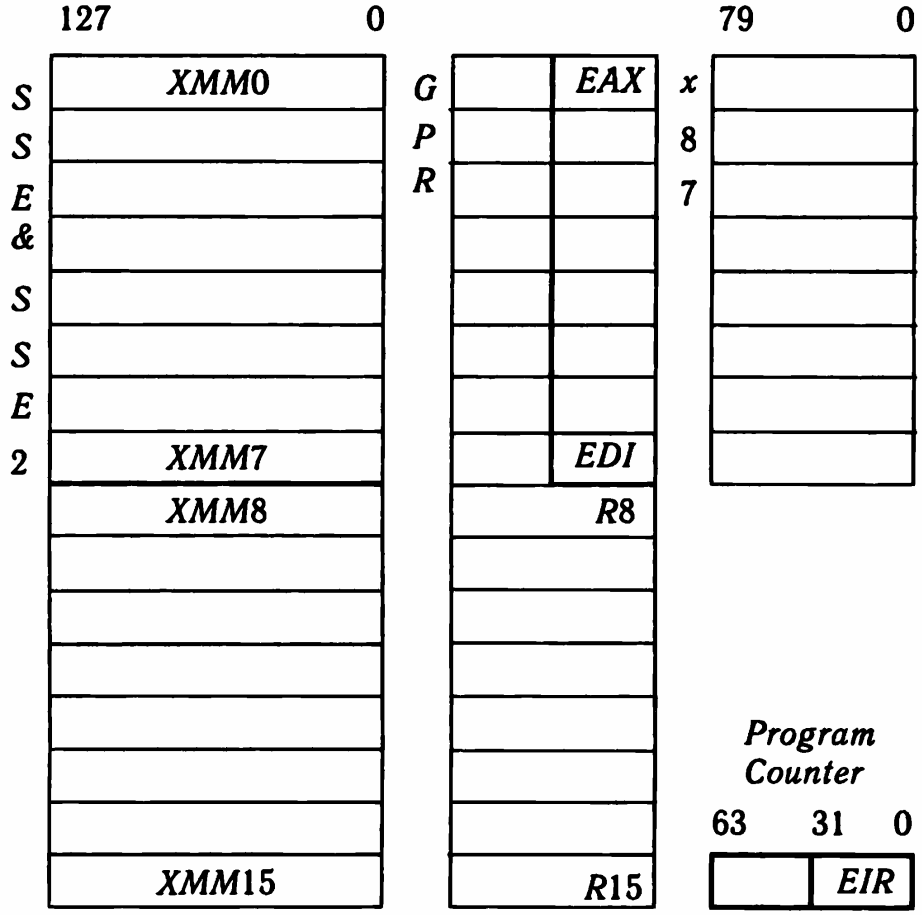


Рис. 3.18. Архітектура *x*86–64

Регістри загального призначення (GPR) доповнені 8 регістрами *R*15–*R*8, що використовуються в 64-бітному режимі, а існуючі регістри *EAX, EBX* розширено з 32 до 64 біт. Вісім нових регістрів додано у блок *SSE*, що забезпечить підтримку *SSE*2. Розширення існуючих регістрів показано на рис. 3.19.

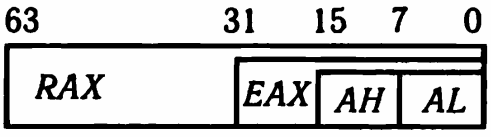


Рис. 3.19. Розширення регістрів загального призначення

Процесор, побудований на основі архітектури $x86$-64, може виконувати існуючі 32-бітні додатки на відміну від того самого процесора Intel Itanium, де систему команд $x86-32$ доводиться моделювати.

Контрольні запитання

1. Назвіть основні характерні ознаки режиму реальної адресації і захищеного режиму.
2. Яка існує кількість ліній шини адреси у реальному і захищеному режимах?
3. Назвіть максимально можливу кількість дескрипторних таблиць, які можна задати.
4. Чим відрізняються тіньові регістри від програмно-доступних?
5. Яку інформацію містять дескрипторні таблиці?
6. Яке призначення регістра *IDTR*.
7. Які команди треба виконати перед переходом у захищений режим роботи?
8. Дайте характеристику існуючих типів циклів шини.
9. Схарактеризуйте поняття «переривання» та «виняток».
10. Чим відрізняється оброблення переривання від оброблення винятку?
11. Яке призначення регістрів, що входять до програмної моделі 32-розрядного процесора?
12. Яке призначення прапорців, що входять до програмної моделі 32-розрядного процесора?
13. Яке призначення регістрів керування і регістрів тестування?
14. Розкажіть про призначення та існуючі типи дескрипторних таблиць.
15. Як значення системного біта визначає тип дескриптора?
16. Дайте визначення процесу свопінгу і поясніть, як він відбувається.
17. Поясніть принцип сторінкової організації пам'яті.
18. Поясніть необхідність і принцип функціонування механізму захисту за привілеями.
19. За яких умов дозволяється зчитувати (записувати) дані певного сегмента?
20. Яка інформація міститься у вентилях викликів?
21. Поясніть особливості багатозадачного режиму роботи.
22. Яке призначення регістра *TR*?
23. Яке призначення пакетного режиму?
24. Поясніть процес пакетного передавання даних.
25. Яке призначення буферів відкладеного запису?
26. В яких випадках може порушуватися порядок обслуговування запитів на запис та зчитування у буферах відкладеного запису?
27. Яке призначення OverDrive-процесорів?
28. Які операційні системи можуть використовуватися у МП Pentium?
29. Назвіть призначення основних блоків структурної схеми процесора Pentium.

30. Поясніть роботу блока передбачення адреси переходу на прикладі виконання команди *JC LABEL*.
31. Яка ефективність поділу кеш-пам'яті на кеш-пам'ять команд та кеш-пам'ять даних?
32. Які засоби виявлення помилок має процесор Pentium?
33. У чому полягає принцип тестування за допомогою функціональної надмірності?
34. Які функції тестування має процесор Pentium?
35. Назвіть призначення та можливості засобів моніторингу продуктивності.
36. Які особливості регістрів-фіксаторів треба враховувати під час розробки програмного забезпечення?
37. Яке призначення та принцип режиму *MMX*?
38. Яке призначення та принцип режиму *SMM*?
39. Як здійснюється перемикання у режим зменшеного енергоспоживання і вихід з нього?
40. Порівняйте архітектури 64-розрядних МП.
41. Поясніть позитивний ефект предикації.
42. Які особливості архітектури *EPIC*?

Розділ 4 ПОБУДОВА МОДУЛІВ ПАМ'ЯТІ МІКРОПРОЦЕСОРНИХ СИСТЕМ

4.1 Класифікація систем пам'яті

Система пам'яті є функціональною частиною мікропроцесорної системи, призначеною для запису, зберігання та видачі інформації. Технічні засоби, що реалізують функції пам'яті, називаються *запам'ятовувальними пристроями (ЗП)*.

Однією з ознак класифікації ЗП є фізична природа середовища, яке використовується для зберігання інформації. За цією ознакою виділяють такі види ЗП: електронні (на пристроях із зарядовим зв'язком — ПЗЗ), магнітні (на циліндричних магнітних доменах), ультразвукові лінії затримки (магнітострикційні, кварцові, зі спеціальних сплавів скла), кріогенні, голографічні.

За швидкістю обміну інформацією з АЛП розрізняють такі типи пам'яті (рис. 4.1): надоперативний ЗП (НОЗП) або регістрова пам'ять МП, внутрішня кеш-пам'ять, зовнішня кеш-пам'ять, оперативна пам'ять (ОЗП), постійна пам'ять (ПЗП), зовнішня пам'ять, або зовнішня запам'ятовувальна пам'ять (ЗЗП).

Надоперативний запам'ятовувальний пристрій, або **регістрова пам'ять мікропроцесора,** — це сукупність регістрів загального призначення. Звернення до НОЗП не потребує від МП виставлення адреси на шину *AB* під час зчитування-запису інформації, тому операції з НОЗП є найбільш швидкодіючими. Тривалість вибірки НОЗП становить 5—7 нс. Загальна кількість 8- або 16-розрядних регістрів у МП, як правило, від 16 до 64.

Внутрішня кеш-пам'ять — оперативний ОЗП статичного типу ємністю 1—16 Кбайт, який вбудовано безпосередньо в МП. Внутрішня кеш-пам'ять працює на тактовій частоті процесора. У моделях *i*386, *i*486 кеш-пам'ять є спільною для даних і команд. У МП Pentium кеш-пам'ять використовується окремо для команд і даних.

Зовнішня кеш-пам'ять так само, як і внутрішня, є ОЗП статичного типу, однак має значно більшу ємність. Її встановлюють на системній платі і вона працює на частоті шини. Зовнішня кеш-пам'ять призначена для зменшення кількості звернень до інших менш швидкодіючих пристроїв пам'яті. Ємність зовнішньої кеш-пам'яті в сучасних ПЕОМ, як правило, 64 Кбайт...1 Мбайт і має тенденцію до збільшення.

Оперативна пам'ять — ОЗП статичного або динамічного типу. В мікросхемах статичного типу елементом пам'яті є тригер на біполярних транзисторах або транзисторах зі структурою метал—діелектрик—напівпровідник (МДН). В ОЗП динамічного типу елементом пам'яті є конденсатор. Оперативна пам'ять припускає зміну свого вмісту під час виконання процесором обчислювальних операцій з даними і може працювати в режимі запису, зчитування і зберігання інформації. Оперативна пам'ять призначена для зберігання змінної інформації — поточних даних, результатів обчислень. Її ємність становить 1...64 Мбайт, тривалість доступу 70—200 нс. Оперативний запам'ятовувальний пристрій є енергозалежним, оскільки інформація, записана в ньому, втрачається після вимкнення живлення.

Постійна пам'ять — спеціальна мікросхема, що містить інформацію, яка не повинна змінюватися у процесі виконання програми. Ця інформація заноситься у ПЗП під час виготовлення або на етапі його програмування у спеціальному пристрої — програматорі, і в процесі роботи мікропроцесорної системи може лише зчитуватися. Постійна пам'ять у МПС працює в режимах зберігання та зчитування і використовується для зберігання таблиць, констант, кодів команд програм, стандартних підпрограм, наприклад, підпрограм *BIOS*,

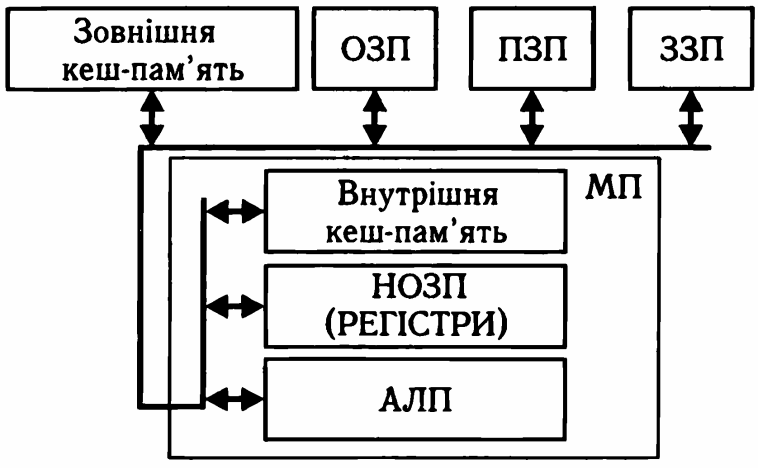


Рис. 4.1. Типи пам'яті в МПС

POS. Як правило, ПЗП має ємність 64...128 Кбайт. Записана в ПЗП інформація зберігається під час вимкнення живлення. Цю властивість ПЗП називають *енергонезалежністю*.

Зовнішня пам'ять реалізується у вигляді накопичувачів зі змінними і незмінними носіями: на твердих і гнучких магнітних дисках, стримерах, оптичних і лазерних компакт-дисках (*CD ROM*). Обмін інформацією з пристроями пам'яті цього типу є найповільнішим, але ця пам'ять є найбільшою за ємністю. Ємність накопичувачів на твердих дисках становить 1...100 Гбайт. Низька швидкодія зовнішніх ЗП на магнітних носіях зумовлюється наявністю електромеханічних пристроїв.

Наявність того чи іншого типу пам'яті поза мікросхемою МП (див. рис. 4.1) зумовлюється функціональним призначенням МПС.

Запам'ятовувальні пристрої характеризуються такими основними параметрами:

розрядністю даних (визначається розрядністю комірки пам'яті);

інформаційною ємністю (визначається кількістю одиниць інформації у бітах, яку ЗП може зберігати одночасно). Інформаційну ємність часто позначають через $N \times n$, де n — розрядність шини даних; N — кількість n-розрядних слів. Так, запис значення інформаційної ємності ЗП 2048×8 або $2К \times 8$ означає, що ЗП може зберігати 2048 байт або 16 384 біт;

тривалістю вибірки (визначається як інтервал від моменту видавання запиту на передавання даних з пам'яті до моменту появи інформації на виході ЗП);

тривалістю циклу звернення (циклу пам'яті) $t_ц$ (визначається мінімально допустимим інтервалом часу між двома послідовними зверненнями до ЗП);

напругою живлення $U_ж$;

потужністю енергоспоживання $P_{сп}$ (визначається добутком струму споживання і напруги джерела живлення). Для деяких типів ЗП наводять два значення потужності енергоспоживання — одне для режиму звернення, коли здійснюється запис або зчитування, інше — для режиму зберігання. Потужність енергоспоживання у режимі зберігання, як правило, істотно менша, ніж потужність енергоспоживання режиму звернення;

питомою вартістю (визначається відношенням вартості ЗП до його інформаційної ємності).

4.2. Побудова модулів постійного запам'ятовувального пристрою

Основною складовою ПЗП є *елемент пам'яті*, який зберігає 1 біт інформації. Елементи пам'яті об'єднані у матрицю накопичувача інформації. Сукупність з n елементів пам'яті, в якій розміщується n-розрядне слово, називають *коміркою пам'яті*, при цьому величина n визначає *розрядність комірки*. Кількість комірок пам'яті дорівнює 2^m, де m — кількість адресних входів, а інформаційна ємність мікросхеми — $2^m \times n$ біт. Для кожної комірки пам'яті є своя адреса. Більшість ПЗП мають словникову організацію, тобто дозволяють паралельне зчитування n розрядів слова $D_{n-1} - D_0$.

Для зчитування інформації з комірки треба на адресні входи мікросхеми ПЗП подати код адреси $A_{m-1} - A0$, які через дешифратор рядків обирають відповідну комірку. Зчитування інформації відбувається за активного (нульового) рівня сигналу $\overline{CS}$. За $\overline{CS} = 1$, виходи $D_{n-1} - D_0$ знаходяться у третьому (високоімпедансному) стані — z-стані. Якщо ЗП має виходи з трьома станами або з відкритим колектором, то вихід ВІС ПЗП може бути з'єднаний безпосередньо із шиною; якщо на виході ВІС ПЗП активних пристроїв не має, використовують *підтягувальні* резистори, вмикання яких забезпечує високий рівень вихідного сигналу; якщо ЗП не має виходів з трьома станами, то слід застосовувати мікросхеми шинних формувачів, наприклад, ВІС *і*8286 або К580ВА86.

За способом програмування, тобто за способом занесення інформації, розрізняють такі типи ПЗП: програмовні маскою одноразово, багаторазово програмовні з ультрафіолетовим стиранням, багаторазово програмовні з електричним стиранням або флеш-пам'ять (див. розд. 10, кн. 2).

Розглянемо побудову модуля ПЗП МПС на базі 8-розрядних процесорів. Схему (рис. 4.2) модуль ПЗП має тоді, коли розрядність шини даних процесора збігається з розрядністю шини даних ПЗП, а інформаційна ємність ПЗП достатня для зберігання інформації.

Нарощування ємності ПЗП проводять тоді, коли необхідна ємність модуля пам'яті перевищує ємність однієї ВІС ПЗП.

Приклад 4.1. Визначити інформаційну ємність, початкову і кінцеву адреси модуля пам'яті МПС 8-розрядного МП. Модуль складається з однієї ВІС К573РФ6, яка з'єднана із системною шиною (див. рис. 4.2).

Оскільки ВІС має 13 адресних входів і 8 виходів даних, її інформаційна ємність становить $2^{13} \times 8 = 8$ Кбайт $\times 8$.

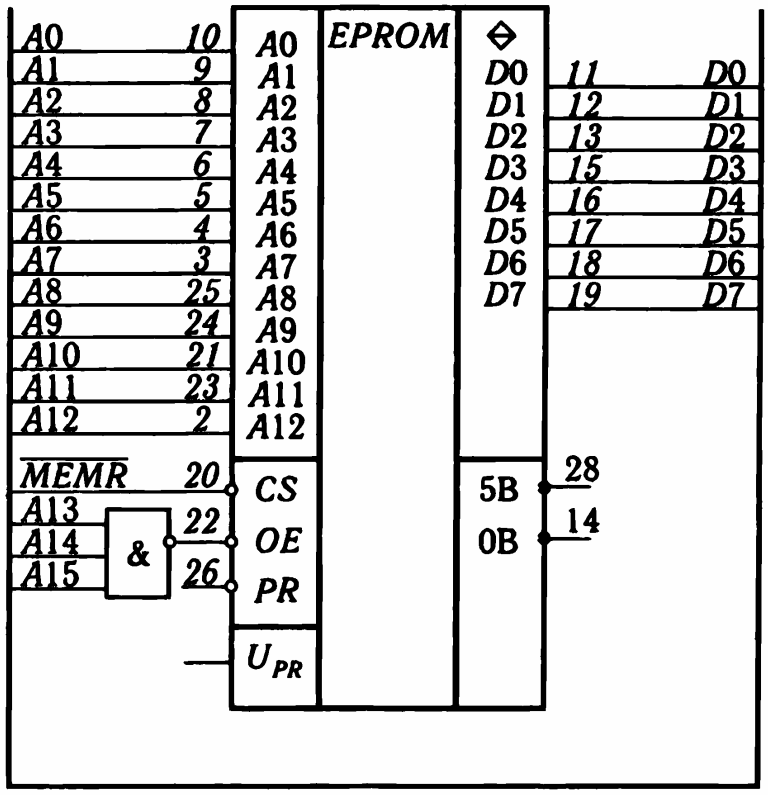


Рис. 4.2. Модуль ПЗП ємністю 8 Кбайт × 8

Для визначення початкової та кінцевої адрес модуля пам'яті зазначимо, що зчитування інформації з ПЗП здійснюється за одночасної дії сигналів $\overline{CS} = 0$ і $\overline{OE} = 0$, при цьому зчитуватиметься вміст комірки з адресою, поданою на входи $A12-A0$. Сигнал $\overline{CS} = 0$ тоді, коли виконується цикл шини ЗЧИТУВАННЯ ПАМ'ЯТІ, тобто $\overline{MEMR} = 0$. Сигнал $\overline{OE} = 0$ в діапазоні адрес з одиничними значеннями розрядів $A13$, $A14$, $A15$. Отже, початкову та кінцеву адреси модуля пам'яті визначають так:

початкова: $1110\ 0000\ 0000\ 0000_2 = 0E000H$;

кінцева: $1111\ 1111\ 1111\ 1111_2 = 0FFFFH$.

Приклад 4.2. Розробити схему модуля ПЗП з інформаційною ємністю 32К × 8 та початковою адресою $8000H$ на базі ВІС К573РФ6. Модуль ПЗП з'єднати із системною шиною 8-розрядного МП. Вибірку окремих ВІС здійснити за допомогою дешифратора.

Для забезпечення інформаційної ємності 32К × 8 схема модуля ПЗП має містити чотири ВІС ПЗП ємністю 8К × 8 кожна (рис. 4.3). Оскільки модуль пам'яті містить чотири ВІС ПЗП, для вибірки кожної з них потрібний дешифратор DC з чотирма виходами a, b, c, d. Щоб початкова адреса модуля ПЗП дорівнювала $8000H$, треба забезпечити вибірку даних з модуля за одиничного значення адресного розряду $A15$ ($8000H = 1000\ 0000\ 0000\ 0000_2$). За нульового значення розряду $A15$ вибірка не здійснюється, тому значення вихідних сигналів дешифратора мають бути одиничними: $a = b = c = d = 1$. Значення розряду $A15$ надходить на вхід дозволу дешифратора E. Після подання на вхід E нульового значення $A15$ жодна з ВІС ПЗП не обирається. Для адресації чотирьох ВІС ПЗП за фіксованого значення старшого адресного

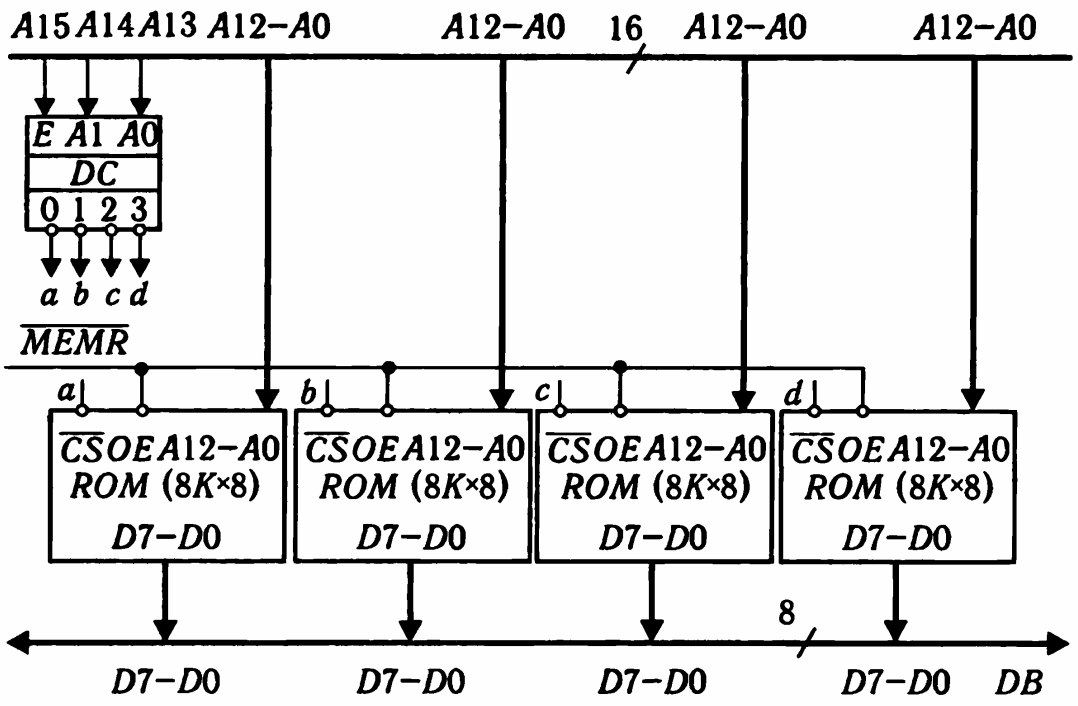


Рис. 4.3. Модуль ПЗП ємністю 32 Кбайт × 8

Таблиця 4.1. **Виходи дешифратора**

*A*15	*A*14	*A*13	*a*	*b*	*c*	*d*
0	*x*	*x*	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	0

розряду *A*15 слід використовувати ще два адресних розряди — *A*14 та *A*13, які надходять на адресні входи дешифратора *A*1 і *A*0. У таблиці відповідності (табл. 4.1) для дешифратора *DC*, що відповідає таким умовам, наведені значення вихідних сигналів *a*, *b*, *c*, *d*, які надходять на входи $\overline{CS}$ чотирьох ВІС для вибірки відповідної ВІС, починаючи з адреси 8000*H*. Символом *x* у табл. 4.2 позначено будь-яке значення вхідного сигналу — 0 або 1.

Молодші 13 розрядів шини адреси (*AB*) подаються на адресні входи *A*12—*A*0 всіх ВІС ПЗП, паралельно адресують комірку всередині однієї ВІС, а два старших розряди *A*14 та *A*13 обирають одну з ВІС ПЗП. Виходи ВІС *D*7—*D*0 з'єднані з шиною даних (*DB*) МПС. Так нарощується ємність модуля.

Розглянемо будову модуля постійної пам'яті для МПС на базі 16-розрядних процесорів, які можуть оперувати як з

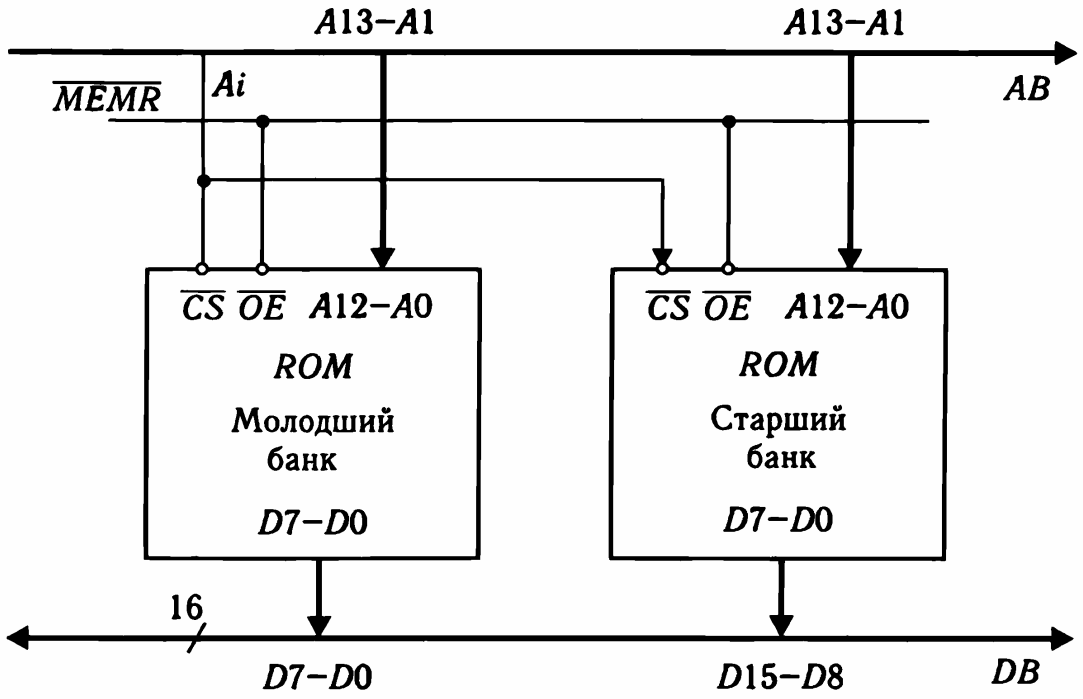


Рис. 4.4. Модуль ПЗП у 16-розрядних МПС

8-, так і з 16-розрядними комірками пам'яті. Для використання 8-розрядних ВІС у модулях пам'яті 16-розрядних процесорів, наприклад, з інформаційною ємністю 1М × 8, постійна пам'ять виконується у вигляді двох *банків* по 512 Кбайт кожний. Один з банків з'єднаний з молодшою половиною шини даних, тобто до розрядів $D7-D0$, і називається *молодшим*, другий — до старшої половини шини даних (розряди $D15-D8$) і називається *старшим*. Молодший банк містить байти з парними адресами ($A0 = 0$), старший — з непарними ($A0 = 1$).

Для адресації байта всередині банку використовують адресні розряди $A19-A1$. Зчитування з ПЗП організовано так, що під час звернення до ПЗП на шину даних МП завжди надходять два байти, тобто зчитується вміст обох банків одночасно. У разі потреби процесор може обирати один необхідний байт з двох. На рис. 4.4 наведено систему пам'яті у вигляді двох банків. Кожний з банків виконано за структурною схемою модуля ПЗП для 8-розрядних процесорів, розглянутих вище.

У мікропроцесорних системах з 32-розрядною шиною даних модуль ПЗП виконується у вигляді чотирьох банків. Інформація зчитується одночасно з усіх чотирьох банків, після чого МП обирає одно-, дво- або чотирибайтове слово залежно від команди, що виконується.

4.3. Побудова модулів оперативного запам'ятовувального пристрою статичного типу

Елементом пам'яті ОЗП статичного типу є тригер. Кожний елемент пам'яті має свою адресу. Для того щоб зчитати інформацію з елемента пам'яті або записати в нього нове значення, його треба обрати поданням на ВІС ОЗП відповідної адреси $A_{m-1} - A_0$. Якщо запам'ятовувальний пристрій допускає звернення до будь-якого елемента пам'яті у довільному порядку, його називають *ОЗП з довільною вибіркою* (*RAM — Random Access Memory* — пам'ять з довільним доступом), див. розд. 9, кн. 2.

Існують ОЗП з інформаційною ємністю $2^m \times 1$ біт (серії К541, К132) і з ємністю $2^m \times 8$ (К537, К581). Для введення інформації в ОЗП першого типу використовують вхід *DI* (*Data Input*), а для виведення — вихід *DO* (*Data Output*). Великі інтегральні схеми ОЗП другого типу мають *n* інформаційних входів-виходів $DIO_{n-1} - DIO_0$. Такий ОЗП припускає читання-запис 8-розрядного коду. Керування режимами (запису, зчитування, зберігання) здійснюється за допомогою сигналів $\overline{CS}$ і $W/\overline{R}$. Одиничний стан сигналу $W/\overline{R}$ визначає режим запису біта інформації в елемент пам'яті, а нульовий — режим зчитування біта інформації з елемента пам'яті.

Розглянемо будову модуля ОЗП статичного типу для МПС з 8-розрядними процесорами, використовуючи ВІС ОЗП з інформаційною ємністю $N \times 1$. Для цього слід нарощувати розрядність даних.

Приклад 4.3. Для МПС з 8-розрядним МП розробити схему модуля ОЗП з інформаційною ємністю 4К × 8 на базі ВІС К537РУ14 з інформаційною ємністю 4К × 1.

Для побудови модуля пам'яті ємністю *4*К × *8* треба вісім ВІС ОЗП К537РУ14, рис. 4.5. Оскільки такі ВІС не мають входу сигналу *OE*, тобто під час записування інформації вони не переходять у *z*-стан, слід використовувати шинний формувач. У цьому разі використаємо шинний формувач КР580ВА86, що з'єднує вихід ВІС із шиною даних під час виконання циклу ЧИТАННЯ ПАМ'ЯТІ, $\overline{MEMR} = 0$ та нульовому рівні сигналів на входах $\overline{W}/R$ і $\overline{CS}$. Кожну з ВІС ОЗП з'єднано з однією з ліній шини даних $D7 - D0$.

Для мікросхем ОЗП, в яких під час записуваня інформації виходи переходять у високоімпедансний стан, шинний формувач не використовується, і виходи ВІС безпосередньо з'єднуються з шиною даних.

Отже, розрядність нарощується під'єднанням однієї ВІС ОЗП до кожної лінії шини даних. При цьому адресні входи для всіх ВІС ОЗП

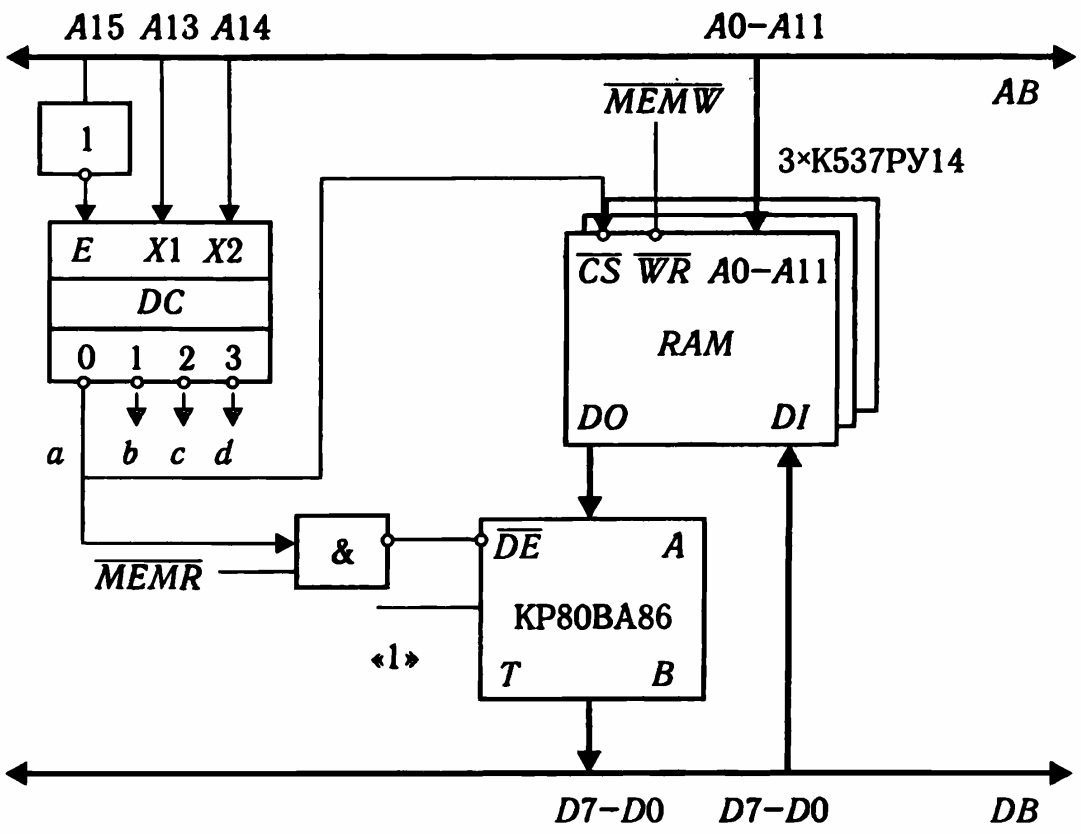


Рис. 4.5. Нарощування розрядності модуля статичного ОЗП

з'єднуються паралельно. Паралельно також з'єднуються входи керування і вибірки. Нарощування ємності ОЗП здійснюється аналогічно нарощуванню ємності ПЗП. На рис. 4.5 показано один з чотирьох можливих варіантів під'єднання модуля ОЗП — до виходу *a* дешифратора, інших трьох модулів ОЗП — до виходів *b*, *c*, *d* дешифратора.

Розглянемо будову модуля оперативної пам'яті для МПС на основі 16-розрядних процесорів. Модуль ОЗП, так само, як і ПЗП, виконується у вигляді двох банків, під'єднаних відповідно до молодшої та старшої половин шини даних.

Приклад 4.4. Розробити схему модуля ОЗП інформаційною ємністю 16К × 16 для МПС із 16-розрядним МП.

У схемній реалізації модуля ОЗП має бути забезпечене читання-запис як 8-, так і 16-розрядних даних. Модуль ОЗП складається з двох банків — молодшого і старшого (рис. 4.6).

Адресні розряди $A13-A1$ з'єднані з адресними входами $A12-A0$ обох банків. Вибірка банків здійснюється одиничним значенням сигналу на виході дешифратора *DC*, сигналами *A*0 для вибірки молодшого банку або $\overline{BHE}$ (*Bus High Enable* — дозвіл старшого байта шини) для вибірки старшого банку. Одиничний рівень сигналу на виході *DC* з'являється після надходження на шину адреси *AB* відповідної адреси ОЗП. Сигнал $\overline{MEMR}$ або $\overline{MEMW}$ вказує на виконання відповідно циклу ЧИТАННЯ ПАМ'ЯТІ або ЗАПИС У ПАМ'ЯТЬ.

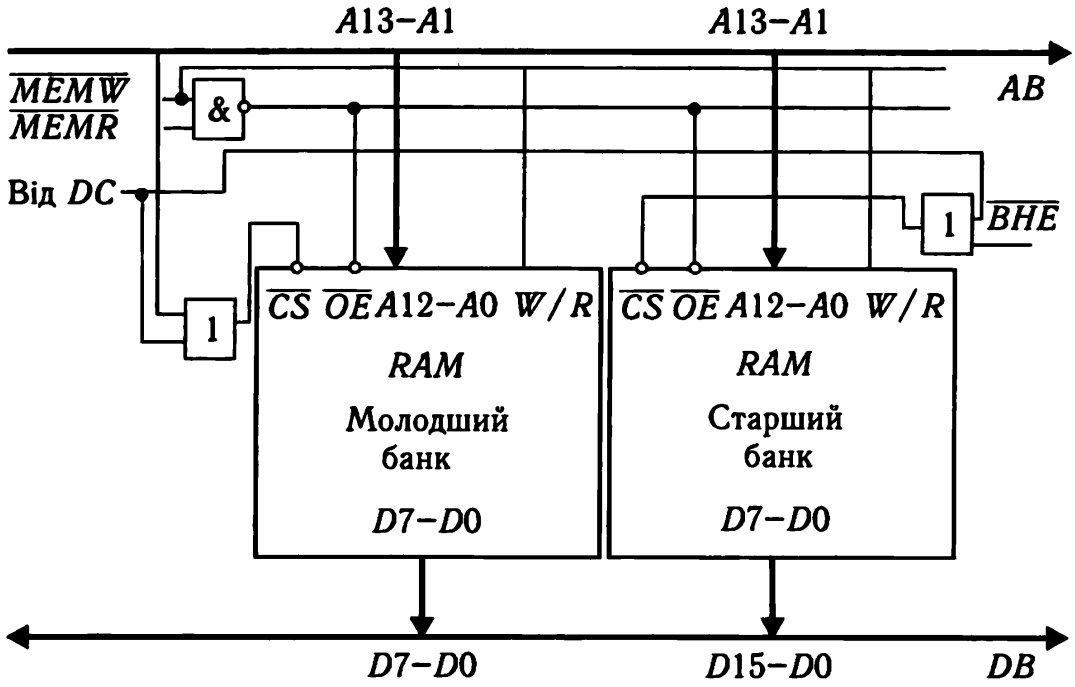


Рис. 4.6. Модуль ОЗП у 16-розрядних МПС

Комбінація значень сигналів $A0$ і $\overline{BHE}$ визначає чотири можливих випадки звернення до пам'яті:

- вибірку байта за парною адресою;
- вибірку байта за непарною адресою;
- вибірку слова за парною адресою;
- вибірку слова за непарною адресою.

Під час *вибірки байта за парною адресою* $A0 = 0$, $\overline{BHE} = 1$. Байт з парною адресою передається по лініях $D7-D0$, тобто здійснюється зчитування або запис байта. У разі запису байта у молодший банк інформація в старшому банку захищена від стирання, тобто одиничне значення сигналу $\overline{BHE}$ забороняє звернення до старшого банку.

Під час *вибірки байта за непарною адресою* $A0 = 1$, $\overline{BHE} = 0$. Наприклад, за командою

MOV BL, BYTE PTR [10001*H*]

вміст комірки пам'яті з адресою *DS:*10001*H* пересилається в молодшу половину 16-розрядного регістра *BX*, тобто у 8-розрядний регістр *BL*. При цьому вміст комірки пам'яті записується у розряди $D15-D8$, тобто старшу половину шини даних, потім, у процесі виконання команди, на молодшу половину внутрішньої 16-розрядної шини МП, а після цього — в регістр *BL*. Цей процес, який називають *маршрутизацією байта*, відбувається автоматично і непомітно для програміста.

Під час *вибірки слова за парною адресою* $\overline{BHE} = 0$, $A0 = 0$. У цьому разі водночас обираються два банки, і 16-розрядне слово передається по лініях $D15-D0$ за один цикл шини.

Якщо слово має непарну адресу, його молодший байт розміщується у старшому банку пам'яті, а старший байт — у молодшому банку. У процесі *вибірки слова за непарною адресою* спочатку $A0 = 1$, $\overline{BHE} = 0$, і по лініях шини $D15-D8$ передається молодший байт. Після цього

генеруються сигнали *A0=0*, $\overline{BHE} = 1$, здійснюється *інкремент* (збільшення на одиницю) повної адреси $A19-A0$, старший байт слова передається з молодшого банку або в молодший банк по лініях шини $D7-D0$. Отже, вибірка слова за непарною адресою потребує два цикли шини. Тому слова доцільно розміщувати за парними адресами, особливо під час організації операцій зі стеком.

4.4. Побудова модулів оперативного запам'ятовувального пристрою динамічного типу

У мікросхемах ОЗП динамічного типу *елементом пам'яті* є конденсатор p–n-переходу МДН-транзистора. Заряджений стан конденсатора вважається станом логічної одиниці, розряджений — станом логічного нуля. Такі елементи пам'яті не можуть тривалий час зберігати свій стан і тому потребують додаткового обладнання для забезпечення періодичного відновлення (регенерації) інформації. Тривалість вибірки для динамічного ОЗП становить 70—200 нс. Порівняно з ОЗП статичного типу ОЗП динамічного типу характеризуються більшою інформаційною ємністю, що зумовлюється меншою кількістю компонент в одному елементі пам'яті, та меншими швидкодією (пов'язано з потребою заряджання і розряджання конденсатора), потужністю споживання і вартістю. Переважно модулі оперативної пам'яті сучасних МПС реалізуються на базі ВІС ОЗП динамічного типу.

Оперативні ЗП динамічного типу (див. розд. 9, кн. 2) працюють у таких режимах: запису, зчитування, зчитування—модифікації—запису, сторінкового запису, сторінкового зчитування, регенерації. Для забезпечення адресування всієї ємності пам'яті використовують мультиплексування адресних сигналів у часі. Спочатку відбувається зчитування або запис даних, адреса яких визначається m молодшими розрядами шини адреси, що супроводжується сигналом строба $\overline{RAS}$ (*Row Address Strobe* — строб адреси рядка). Після цього відбувається зчитування або запис даних, адреса яких визначається m старшими розрядами шини адреси, що супроводжується сигналом строба $\overline{CAS}$ (*Column Address Strobe* — строб адреси стовпця). Зчитування інформації здійснюється за заднім фронтом сигналу $\overline{CAS}$ за $\overline{W}/R = 1$, запис — за заднім фронтом сигналу $\overline{CAS}$ за $\overline{W}/R = 0$. Режим зчитування—модифікації—запису полягає у зчитуванні інформації з подальшим записом в один і той самий елемент пам'яті. Сторінкові режими запису і зчитування реалізуються звер-

ненням до ВІС за адресою рядка з вибіркою елемента пам'яті цього рядка зміною адрес стовпців. Регенерація інформації здійснюється зверненням до кожного з рядків, при цьому формується адреса рядка і сигнал $\overline{RAS}$, а сигнал $\overline{CAS}$ дорівнює логічній одиниці. Процес регенерації припиняється після звернення МП до ОЗП. У цьому випадку обробляється вимога МП, після чого режим регенерації триває з тієї адреси, на якій він був припинений.

Для керування ОЗП динамічного типу використовують контролери динамічної пам'яті, наприклад, К1810ВТ03, К1810ВТ02, *i*8207, які формують адресні та керуючі сигнали у режимах роботи і регенерації, а також здійснюють арбітраж, тобто розв'язання конфліктів між запитами на регенерацію і звернення до пам'яті. Контролер формує також сигнал готовності блока динамічної пам'яті до обміну.

Контролер динамічної пам'яті *i*8207 (рис. 4.7) призначений для керування чотирма ВІС ОЗП динамічного типу по 512 Кбайт кожний. Він складається з адресних буферів *B*1 і *B*2, лічильника рядків регенерації (ЛРР), двох мультиплексорів *M*1 і *M*2 та системи керування. Система керування аналізує вхідні сигнали і здійснює керування всіма блоками контролера.

Під час виконання команд читання-запису пам'яті контролер забезпечує з'єднання з виходами *AO*8—*AO*0 спочатку з молодшою, а потім зі старшою половинами адреси, формування стробувальних сигналів $\overline{RAS}$ і $\overline{CAS}$ для кожної з чотирьох ВІС пам'яті, а також сигналу $\overline{WE}$ (*Write Enable* —

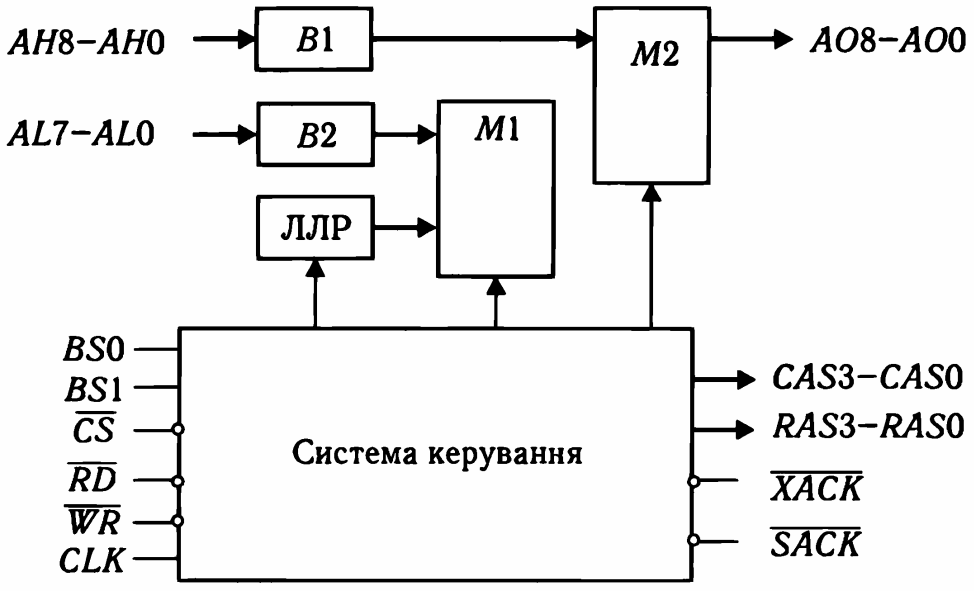


Рис. 4.7. Структурна схема контролера динамічної пам'яті *i*8207

Таблиця 4.2. **Вибір ВІС динамічної пам'яті**

Сигнал		Блок пам'яті
*BS*1	*BS*0	
0	0	0
0	1	1
1	0	2
1	1	3

дозвіл запису). У режимі регенерації контролер забезпечує з'єднання з виводами $AO8-AO0$ виходів лічильника рядків регенерації і формування сигналу $\overline{RAS}$.

Вибір однієї з чотирьох ВІС пам'яті здійснюється сигналами $BS1-BS0$ (табл. 4.2).

Контролер формує сигнал $\overline{XACK}$ наприкінці циклу читання-запису, вказує на закінчення циклу взаємодії з центральним процесором та сигнал $\overline{SACK}$ на початку циклу звернення до пам'яті.

Використання цих сигналів показано на прикладі під'єднання контролера *i*8207 із системною шиною 16-розрядного процесора (рис. 4.8).

На виході $\overline{SACK}$ формується сигнал логічного нуля тоді, якщо контролер виконує такти регенерації інформації в динамічному ОЗП. Сигнал $\overline{XACK}$ використовується як стробувальний сигнал для керування шинними формувачами модуля пам'яті. У цьому прикладі використано два шинних

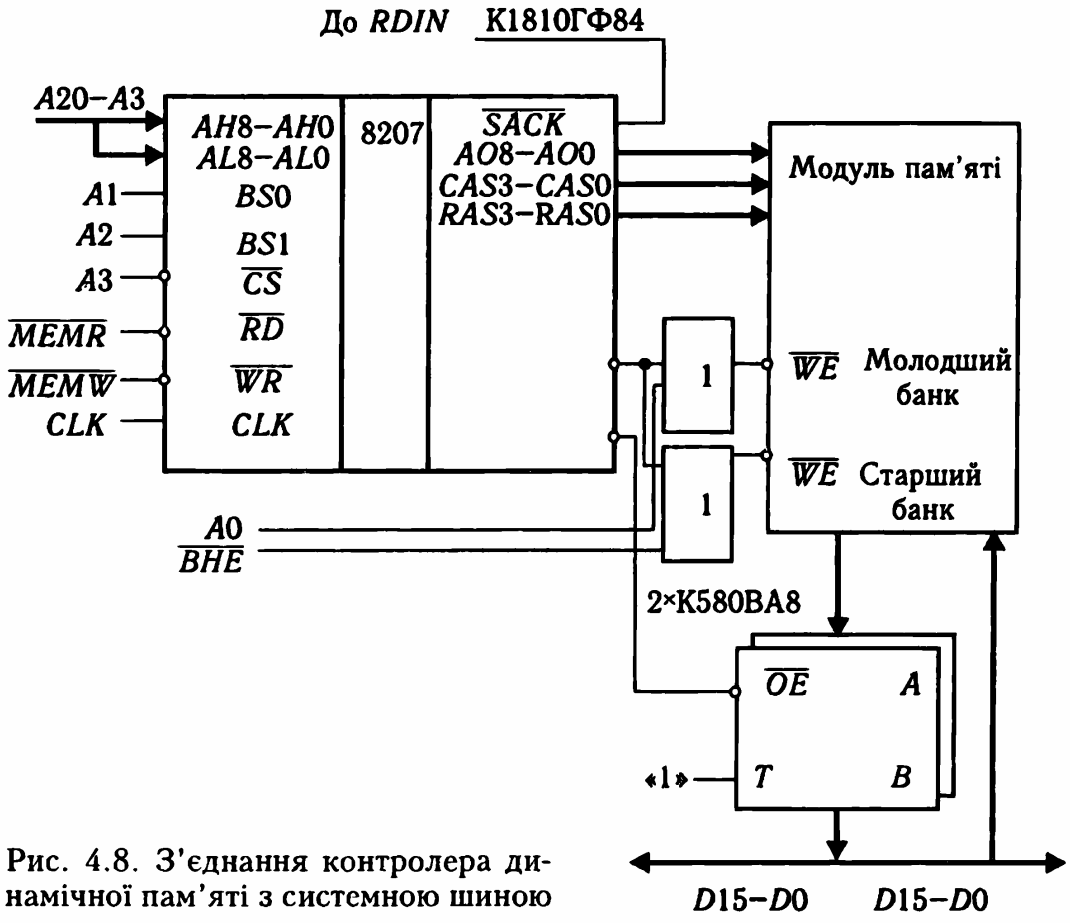


Рис. 4.8. З'єднання контролера динамічної пам'яті з системною шиною

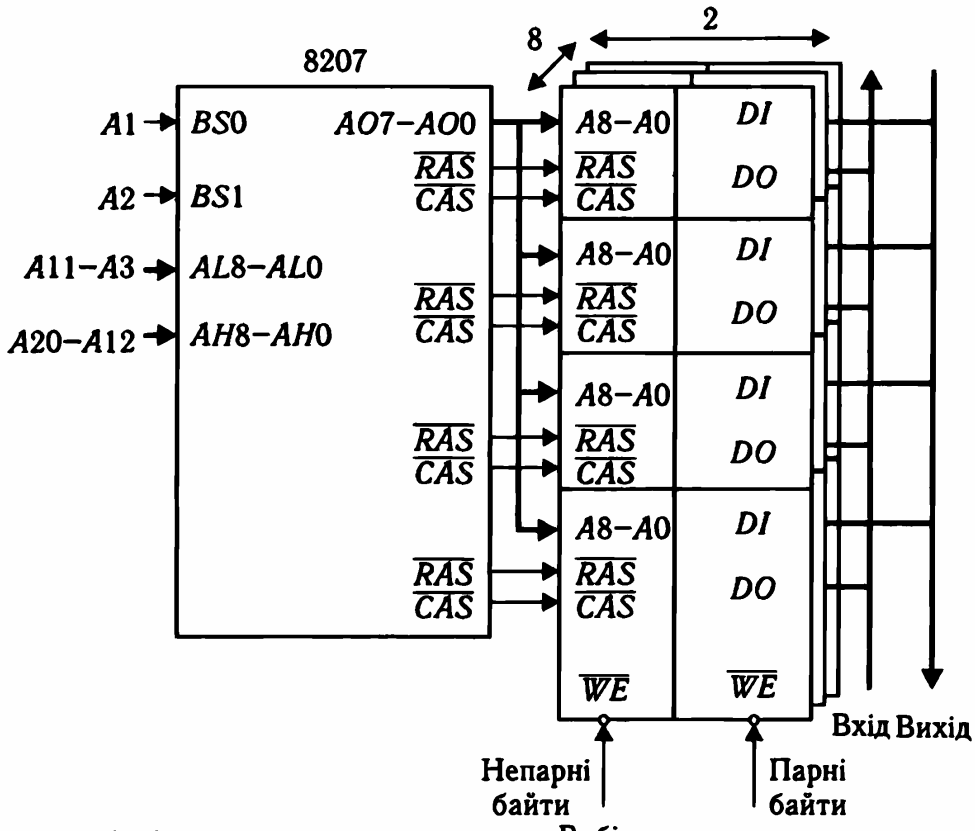


Рис. 4.9. Система динамічної пам'яті

формувачі К580ВА86 (*i*8286), оскільки шина даних є 16-розрядною.

Схему модуля динамічної пам'яті 16-розрядного процесора зображено на рис. 4.9.

Контролер динамічної пам'яті фактично виконує роль інтерфейсу модуля пам'яті та шини процесора. Динамічну пам'ять організовано у вигляді двох банків, кожний з яких складається з чотирьох блоків ємністю по 512 Кбайт. Загальна ємність пам'яті дорівнює 2 Мбайт. Шина адреси процесора з'єднана з контролером. Лінії $A1$ і $A0$ під'єднуються до входів вибирання банку $BS0$, $BS1$, біти $A11-A3$ є адресою рядка, а біти $A20-A12$ — адресою колонки динамічної пам'яті.

4.5. Принципи організації кеш-пам'яті

Збільшення ємності пам'яті МПС зумовлює зниження швидкодії операцій обміну інформацією між процесором і модулем пам'яті. Навіть за час звернення до пам'яті, що дорівнює 70 нс, неможливо отримати потрібну інформацію за один цикл шини. Це призводить до необхідності виконання тактів

очікування у процесі роботи процесора для того, щоб тривалість звернення до пам'яті була узгоджена з тривалістю виконання команди у процесорі. Підвищення швидкодії обміну інформацією можливе через реалізацію додаткової пам'яті порівняно невеликої ємності, звернення до якої здійснюється на тактовій частоті процесора. Таку пам'ять називають *кеш-пам'яттю,* або *буферною пам'яттю.* Кеш-пам'ять реалізується на базі ВІС ОЗП статичного типу. Інформаційна ємність і принцип організації кеш-пам'яті є одними з основних чинників, що визначають продуктивність роботи МПС.

Кеш-пам'ять використовують не лише для обміну даними між МП та ОЗП, а й для обміну між ОЗП і зовнішніми накопичувачами. В основу роботи кеш-пам'яті покладено принципи часової та просторової локальностей програм.

Принцип часової локальності полягає в тому, що під час зчитування будь-яких даних з пам'яті існує висока ймовірність звернення програми впродовж деякого невеликого проміжку часу знову до них. *Принцип просторової локальності* ґрунтується на високій імовірності того, що програма через деякий проміжок часу звернеться до комірки пам'яті, наступної за тією, до якої вона перед цим зверталася.

Згідно з принципом часової локальності інформацію у кеш-пам'яті краще зберігати впродовж деякого часу, а принцип просторової локальності вказує на доцільність розміщення у кеш-пам'яті вмісту кількох сусідніх комірок, тобто певного блока інформації. Лінійні ділянки програм (без переходів) здебільшого не перевищують 3—5 команд, тому нераціонально використовувати блоки інформації, ємність яких перевищує ємність пам'яті, потрібну для зберігання 3—5 команд. Як правило, інформація з основної пам'яті завантажується у кеш-пам'ять блоками по 2—4 слова і зберігається там деякий час.

У разі звернення процесора до пам'яті спочатку перевіряється наявність у кеш-пам'яті даних, які запитуються, і якщо їх немає, здійснюється завантаження у кеш-пам'ять потрібної інформації. Правильна організація роботи кеш-пам'яті забезпечує підвищення швидкодії системи, оскільки переважно відбувається звернення процесора до кеш-пам'яті, а не до повільнішої основної оперативної пам'яті.

Залежно від способу відображення інформації з основної пам'яті на кеш-пам'ять розрізняють такі типи кеш-пам'яті:

- кеш-пам'ять з прямим відображенням;
- повністю асоціативна кеш-пам'ять;
- множинна асоціативна кеш-пам'ять.

Кеш-пам'ять з прямим відображенням є найпростішим типом кеш-пам'яті (рис. 4.10). Кеш-пам'ять містить дві частини — кеш-пам'ять даних і кеш-пам'ять ознак. Припустимо, що ємність ОЗП МПС становить 4 Гбайт. Ця ємність розбивається на 64 × 1024 рівних частин по 64 Кбайт. Блок даних ємністю 4 байт пересилається з кожної із цих частин оперативної пам'яті в один 32-розрядний рядок кеш-пам'яті даних. Ємність кеш-пам'яті даних становить 64 Кбайт, тому кількість рядків дорівнює 64 Кбайт/4 байт = 16 × 1024. Отже, під кожну з 64 × 1024 частин ОЗП у кеш-пам'яті відводиться один рядок: 32-розрядна адреса чотирибайтового блока в ОЗП поділяється на дві частини. Молодші 16 розрядів адреси $A15-A0$ називають *індексом*, старші 16 розрядів $A31-A16$ — *ознакою*. Ознака пересилається у кеш-пам'ять ознак, яка містить 16 · 1024 рядків і має загальну ємність 32 Кбайт.

Для визначення адреси одного рядка кеш-пам'яті ознак треба 14 адресних розрядів $A15-A2$. Для визначення адреси одного рядка кеш-пам'яті даних також необхідно 14 адресних розрядів $A15-A2$, а для визначення одного байта у рядку — 2 розряди $A1$, $A0$.

За потреби зчитування даних з пам'яті процесор звертається спочатку до кеш-пам'яті та перевіряє, чи містить вона необхідні дані. Цей процес відбувається порівнянням ознаки, записаної у кеш-пам'яті ознак, з 16 старшими розрядами адреси, яку процесор виставляє на 32-розрядну шину адреси. Збіг цих величин означає, що у кеш-пам'яті зберігаються необхідні дані. У цьому разі ці дані зчитуються з кеш-пам'яті. Якщо величини не збігаються, то виконується копіювання відповідних даних з оперативної пам'яті у кеш-пам'ять.

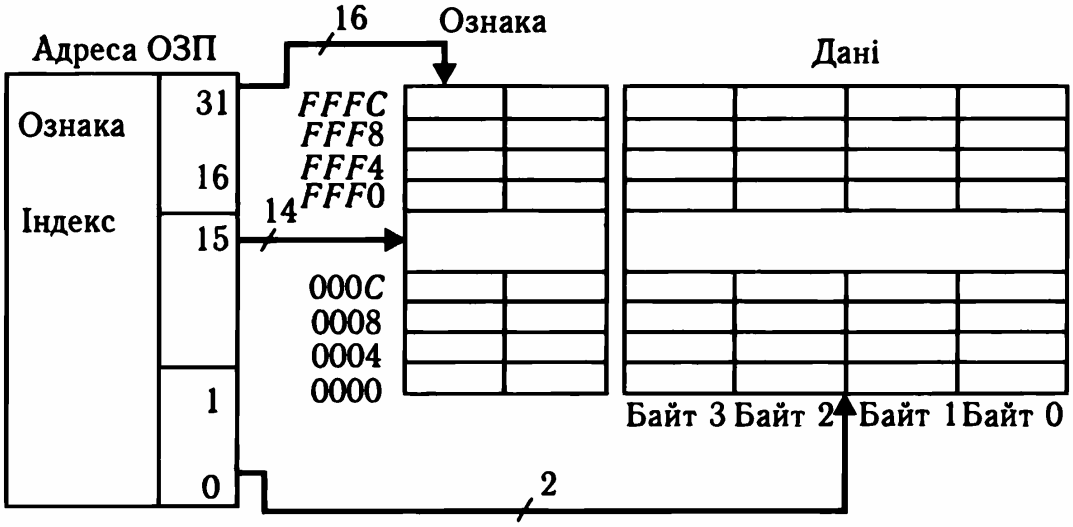


Рис. 4.10. Структурна схема кеш-пам'яті з прямим відображенням

Перевагою цього типу кеш-пам'яті є порівняно висока швидкодія, що пояснюється потребою лише в одному порівнянні ознаки зі старшими розрядами адреси ОЗП.

Недоліком кеш-пам'яті з прямим відображенням є виникнення конфліктів у разі, якщо старші 16 розрядів адреси, виставленої процесором, збігаються з ознакою, записаною у кеш-пам'яті ознак, а індекси необхідного блока та рядка у кеш-пам'яті даних не збігаються. Це означає, що рядок у кеш-пам'яті даних, відведений для цієї частини ОЗП, яка визначається старшими 16 адресними розрядами, вже зайнятий. У цьому випадку вміст рядка кеш-пам'яті даних пересилається назад у ОЗП, а в рядок пересилається необхідний чотирибайтовий блок, внаслідок чого збільшується кількість пересилань між кеш-пам'яттю та ОЗП та тривалість обміну інформацією.

Структурну схему повністю асоціативної кеш-пам'яті показано на рис. 4.11.

У цій схемі усунено недолік кеш-пам'яті з прямим відображенням, оскільки будь-який блок ОЗП може відображатися у будь-якому рядку кеш-пам'яті. У кеш-пам'ять ознак записується 30-розрядна ознака, тобто старші 30 розрядів $A31$ — $A2$ адреси чотирибайтового блока ОЗП. У рядок кеш-пам'яті даних записується чотирибайтовий блок.

Якщо кеш-пам'ять не заповнена, блок записується у будь-який вільний рядок, а якщо кеш-пам'ять заповнена, блок з ОЗП записується у той рядок кеш-пам'яті даних, до якого було найменше звернень. Недоліки кеш-пам'яті з прямим відображенням усуваються за рахунок додаткового обладнання, призначеного для визначення блока, до якого було найменше звернень. При цьому також збільшується тривалість

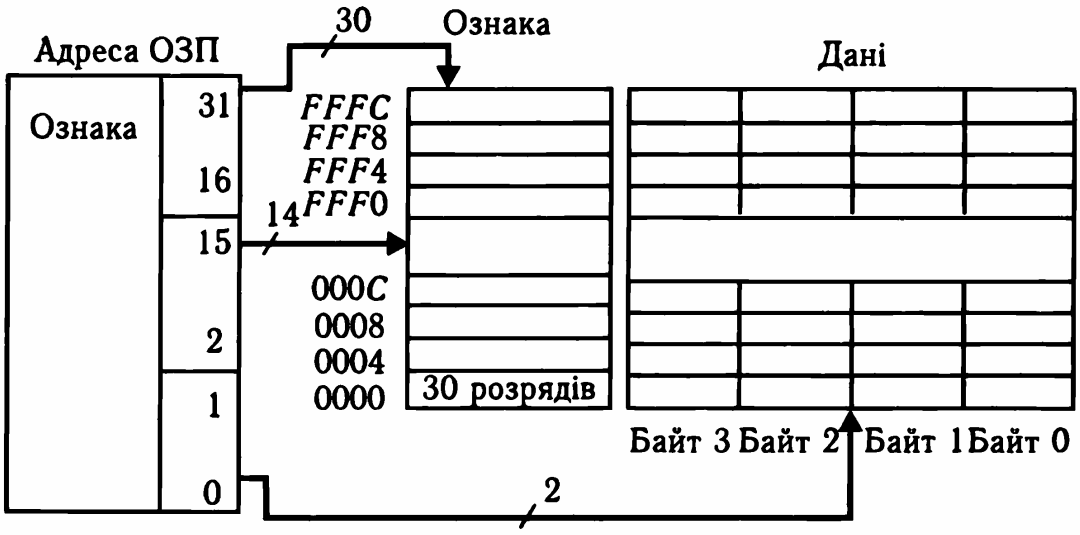


Рис. 4.11. Структурна схема повністю асоціативної кеш-пам'яті

оброблення запитів через необхідність порівняння 30-розрядної адреси та ознаки, записаної у кеш-пам'яті ознак. Максимальна кількість таких порівнянь становить 16×1024.

Множинна асоціативна кеш-пам'ять поєднує переваги обох попередніх типів. Рядки цієї кеш-пам'яті об'єднані в групи по 2, 4 і більше (згідно з цим розрізняють дво-, чотири- і більше входову множинну асоціативну кеш-пам'ять). Структурну схему двовходової множинної асоціативної кеш-пам'яті зображено на рис. 4.12.

Кеш-пам'ять даних складається з 8×1024 груп, кожна з яких містить два рядки. Індекс, тобто 16 молодших розрядів адреси ОЗП, адресує байт у кеш-пам'яті даних (13 розрядів адресують одну з 8×1024 груп, 1 розряд — рядок у групі та 2 розряди — байт у рядку). Ознака, тобто 16 старших адресних розрядів, записується в рядок пам'яті ознак. Тому для блоків з одним і тим самим індексом відводиться два рядки буфера. Отже, якщо один з рядків групи зайнято деяким блоком, то наступний блок з таким самим індексом розміщується у вільний рядок. Всередині групи кеш-пам'ять є повністю асоціативною. Кількість порівнянь адрес ОЗП з ознаками дорівнює двом.

Зростання ємності кеш-пам'яті (тобто кількості рядків у групі) збільшує ефективність її роботи, але зростає кількість порівнянь адреси, отже, тривалість оброблення запиту комірки ОЗП. Ефективність роботи кеш-пам'яті характеризується *коефіцієнтом удалих звернень*. Кеш-пам'ять з прямим відображенням інформаційною ємністю $N \times n$ має такий самий коефіцієнт вдалих звернень, що й двовходова множинна асоціативна кеш-пам'ять ємністю $(N \times n)/2$.

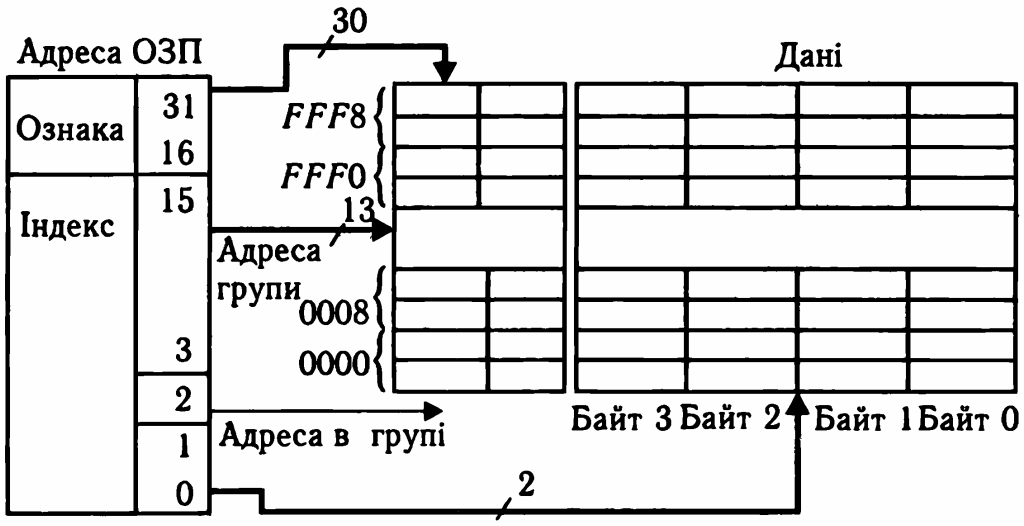


Рис. 4.12. Структурна схема двовходової множинної асоціативної кеш-пам'яті

Усі розглянуті типи кеш-пам'яті мають властивість зберігати окремі копії інформації, яка міститься в основній пам'яті. Під час запису в кеш-пам'ять може порушуватися цілісність збережуваних даних, тобто вміст кеш-пам'яті перестане відповідати вмісту ОЗП. Існує кілька способів відновлення інформації в ОЗП, основними з яких є спосіб наскрізного та спосіб зворотного записів.

Спосіб наскрізного запису полягає в тому, що інформація записується як у кеш-пам'ять, так і в ОЗП. *Спосіб зворотного запису* передбачає запис інформації в ОЗП лише тоді, якщо вона змінюється в кеш-пам'яті. Кожному рядку кеш-пам'яті зіставляється спеціальний біт — біт запису, що вказує на зміну вмісту рядка. Під час заміщення рядка кеш-пам'яті новим блоком інформації з ОЗП перевіряється стан біта запису, і якщо біт установлено, то виконується перезапис блока з кеш-пам'яті в ОЗП. Лише після цього в кеш-пам'яті розміщується новий блок з ОЗП. Цей спосіб є ефективнішим, оскільки дає змогу зменшити кількість звернень до ОЗП.

Правильне розміщення даних в ОЗП сприяє раціональній організації роботи програмного забезпечення та підвищенню швидкодії роботи МПС, оскільки, по-перше, пов'язані між собою дані доцільно розміщувати у найближчих комірках ОЗП. У цьому разі під час завантаження блока даних у кеш-пам'ять існує висока ймовірність того, що після оброблення першого слова процесор обиратиме друге слово з кеш-пам'яті, а не з ОЗП, що дасть змогу ефективніше використовувати кеш-пам'ять. По-друге, під час запису слід вирівнювати дані в ОЗП по межі рядка кеш-пам'яті. Припустимо, що програма обробляє трибайтове слово, а довжина рядка кеш-пам'яті дорівнює 4 байт. Якщо розмістити слова в ОЗП підряд (рис. 4.13), то частини одного трибайтового слова, наприклад, слів *D*2 та *D*3, будуть розміщені у сусідніх рядках кеш-пам'яті.

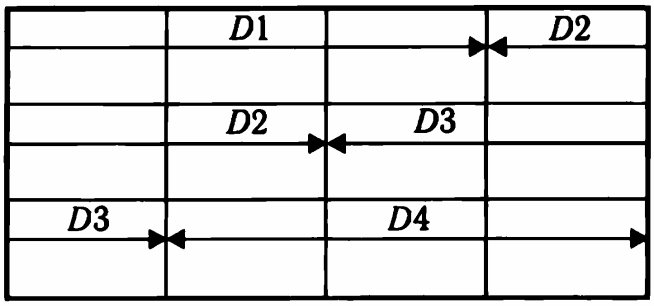


Рис. 4.13. Розміщення трибайтових слів без вирівнювання по межі блока

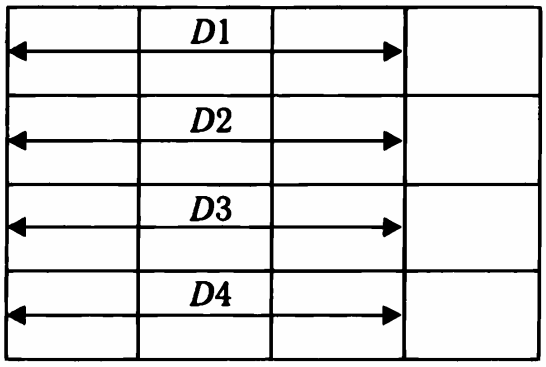


Рис. 4.14. Розміщення трибайтових слів з вирівнюванням по межі чотирибайтового рядка кеш-пам'яті

Без вирівнювання даних досить часто трапляються невдалі звернення до кеш-пам'яті. На рис. 4.14 показано запис трибайтових слів $D1 - D4$, вирівняних по межі чотирибайтового рядка кеш-пам'яті. Це дає змогу значно зменшити кількість невдалих звернень до кеш-пам'яті.

4.6. Принципи організації стекової пам'яті

Стековою пам'яттю, або *стеком*, називають пам'ять, в якій реалізовано принцип: останній увійшов — перший вийшов (*LIFO — Last Input First Output*), тобто дані, записані останніми, зчитуються першими. У МПС стекова пам'ять використовується для викликів підпрограм, в тому числі і вкладених, та оброблення переривань.

За способом реалізації розрізняють апаратний і апаратно-програмний стеки.

Апаратний стек — сукупність регістрів, зв'язок між якими організовано так, що під час записування і зчитування даних вміст стеку автоматично зсувається. Принцип роботи апаратного 8-рівневого стеку наведено на рис. 4.15.

Під час записування *слова* 1 у стек воно розміщується у першій вільній комірці пам'яті (у першому регістрі) — *вершині стеку*. Наступне слово зсуває попереднє на одну комірку вгору, займає його місце і т. д. Запис *слова 9* призводить до переповнення стеку і втрати *слова 1*. Зчитування слів зі стеку здійснюється у зворотному порядку, тобто спочатку зчитується *слово 9*, що записано останнім. Зчитування відбувається у зворотному порядку, наприклад, зчитування *слова* 6 неможливе, доки не будуть зчитані *слова 7, 8, 9*.

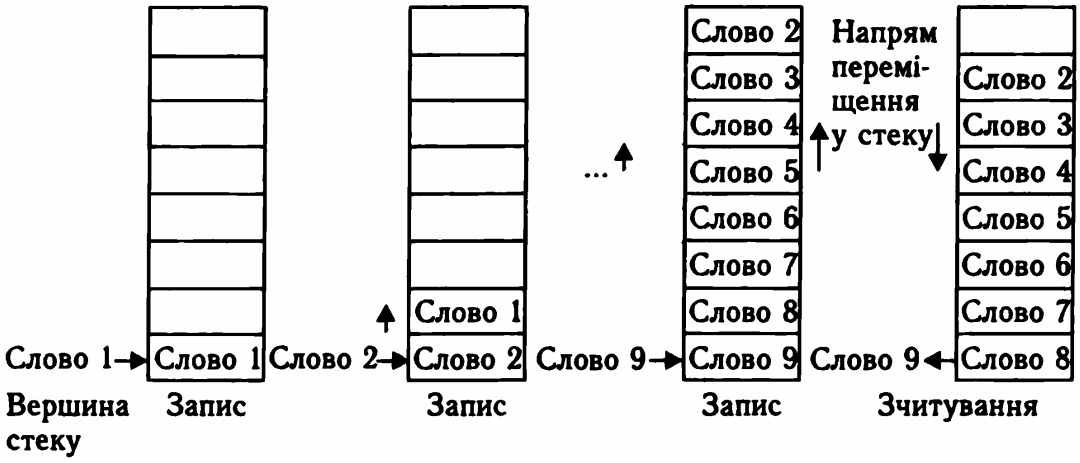


Рис. 4.15. Принцип роботи апаратного стеку

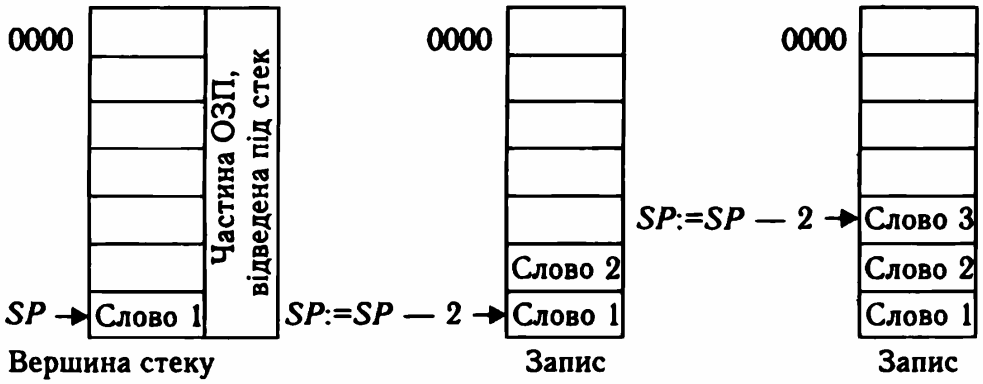


Рис. 4.16. Принцип роботи апаратно-програмного стеку

Інформаційна ємність апаратного стеку визначається як $N \times n$, де кількість n-розрядних слів N дорівнює кількості регістрів, яка може бути кілька десятків. Апаратні стеки, що застосовуються у *PIC*-процесорах, мають 2, 8 або 16 регістрів (N = 2,8,16), в яких розміщуються 12-, 14-, 16-розрядні слова (n = 12,14,16). Основною перевагою апаратного стеку є висока швидкодія, а недоліком — обмежена інформаційна ємність.

Апаратно-програмний стек реалізується через використання частини ОЗП статичного типу та спеціального регістра *SP* (*Stack Pointer* — покажчик стеку), який містить адресу останньої зайнятої комірки стеку. Принцип роботи апаратно-програмного стеку для МП мікропроцесорів 80 × 86 показано на рис. 4.16. В апаратно-програмному стеку під час запису і зчитування фізичного зсуву даних не відбувається. Зсув даних буває після зміни значення регістра *SP*. На початку програми в регістр *SP* заносять адресу вершини стеку. Після кожної операції запису (зчитування) вміст регі-

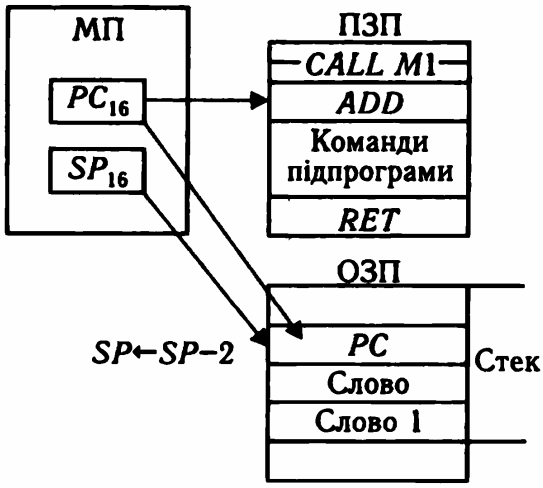


Рис. 4.17. Робота стеку під час виклику підпрограм

стра *SP* змінюється. Для МП 80 × 86 одночасно можна записувати у стек або зчитувати з нього двобайтові слова, тому значення регістра *SP* змінюється на два.

Під час запису в стек значення регістра *SP* зменшується на два (стек «зростає» в область малих адрес), а під час зчитування зі стеку — збільшується на два. Отже, покажчик стеку *SP* завжди містить адресу комірки, до якої відбулося останнє звернення. У деяких командах, наприклад, у командах викликів підпрограм *CALL*, переривань *INT*, поверненнь з підпрограм *RET*, звернення до стеку здійснюється автоматично. Під час виклику підпрограми (рис. 4.17) у стеку запам'ятовується адреса команди, наступної після виклику команди *ADD*, тобто вміст програмного лічильника *PC* запам'ятовується у верхній незайнятій комірці стеку, а покажчик стеку зменшується на два.

Після повернення з підпрограми за командою *RET* вміст верхньої комірки стеку перезаписується у програмний лічильник *PC*, покажчик стеку *SP* збільшується на два. Після цього починає виконуватися команда *ADD*. Крім команд *CALL*, *INT* і *RET*, для роботи зі стеком використовуються також команди *PUSH* і *POP*, призначені для тимчасового запам'ятовування у стек умісту регістрів і відновлення, тобто пересилання інформації зі стеку в регістри. У МП Intel, починаючи з МП *i*286, існують команди *PUSHA* і *POPA* (*PUSH All* і *POP All*), призначені для тимчасового запам'ятовування у стеку і відновлення вмісту всіх регістрів МП. До апаратно-програмного стеку можна звернутися також як до ОЗП з довільною вибіркою. У МП *i*80 × 86 для цього використовують непряму адресацію за допомогою регістра *BP*. Тому в стек можна записати значення параметрів підпрограм перед їх безпосереднім викликом.

Використання стекової пам'яті дає змогу підвищити швидкодію МПС, зменшуючи тривалість однієї з найповільніших операцій — звернення до зовнішньої пам'яті.

Контрольні запитання

1. Дайте визначення і наведіть приклади енергозалежних та енергонезалежних ЗП.
2. Які є типи систем пам'яті?
3. Порівняйте за швидкодією типи систем пам'яті.
4. Визначте інформаційну ємність у бітах та кількість ліній шини даних для ЗП, позначених: а) 1024 × 8; б) 4048 × 16.
5. Як визначити питому вартість ЗП?
6. Для чого призначений ПЗП?
7. Як виконано елементи пам'яті в ПЗП різних типів?
8. Дайте визначення комірки пам'яті.
9. Назвіть типи ПЗП.
10. Як здійснюється занесення інформації у ВІС ПЗП, програмовані маскою?
11. Як заноситься інформація у ВІС однократно програмованого ПЗП?
12. Як здійснюється занесення та стирання інформації у ПЗП *EPROM*?
13. Як здійснюється запис інформації у флеш-пам'ять і її стирання?
14. Наведіть приклад нарощування ємності ПЗП удвічі?
15. Поясніть поняття банку пам'яті.
16. Наведіть приклад побудови модуля ПЗП у 16-розрядній МПС на базі 8-розрядних ВІС ПЗП.
17. Що таке елемент пам'яті статичного ОЗП?
18. Що таке одно- та багаторозрядна організації матриці накопичувача ОЗП?
19. Назвіть основні параметри ОЗП статичного типу.
20. Як співвідносяться значення потужності у режимах читання-запису інформації та режимі зберігання інформації?
21. Як здійснюється нарощування розрядності у модулі статичного ОЗП?
22. Які особливості має побудова модулів оперативної пам'яті для МПС на базі 16-розрядних процесорів?
23. Які сигнали використовуються для вибірки банків пам'яті ОЗП?
24. Назвіть чотири можливих випадки звернення до пам'яті у 16-розрядних процесорах?
25. Що таке маршрутизування байта?
26. Які рекомендації можна дати щодо розміщення даних у стеку?
27. Що таке елемент пам'яті динамічного ОЗП?
28. Назвіть основні параметри ОЗП динамічного типу.
29. Порівняйте параметри ОЗП статичного і динамічного типів. Назвіть недоліки і переваги ОЗП динамічного типу.
30. Схарактеризуйте способи адресування ОЗП динамічного типу.
31. Що таке режим регенерації пам'яті?
32. Які функції виконує контролер динамічної пам'яті?
33. Поясніть призначення сигналів $\overline{RAS}$ і $\overline{CAS}$.
34. Наведіть визначення і призначення кеш-пам'яті.
35. У чому полягає принцип часової локальності?
36. У чому полягає принцип просторової локальності?

37. Поясніть принцип дії кеш-пам'яті з прямим відображенням.
38. Поясніть принцип дії повністю асоціативної кеш-пам'яті.
39. Поясніть принцип дії множинної асоціативної кеш-пам'яті.
40. Порівняйте два види кеш-пам'яті з прямим відображенням та повністю асоціативну.
41. Порівняйте повністю асоціативну кеш-пам'ять з множинною асоціативною кеш-пам'яттю.
42. Як відновлюється інформація в ОЗП за способом наскрізного запису?
43. Як відновлюється інформація в ОЗП за способом зворотного запису?
44. Дайте визначення стекової пам'яті.
45. Яке призначення стекової пам'яті?
46. Поясніть принцип дії апаратного стеку.
47. Поясніть принцип дії апаратно-програмного стеку.
48. Дайте порівняльну характеристику апаратного та апаратно-програмного стеків.
49. Поясніть призначення регістра *SP*.
50. Які операції виконує процесор за командами *CALL*, *RET*, *PUSH*, *POP*, *PUSHA*, *POPA*?

Розділ 5

ІНТЕРФЕЙС ПРИСТРОЇВ ВВЕДЕННЯ-ВИВЕДЕННЯ

5.1. Функції інтерфейсу введення-виведення

Одним з найважливіших завдань проектування МПС є організація взаємодії із зовнішніми пристроями — джерелами і приймачами даних. Прикладами пристроїв введення-виведення (ПВВ), що є як джерелами, так і приймачами інформації, є накопичувачі на гнучких і твердих магнітних дисках. До пристроїв введення належать перемикачі, клавіатура, аналого-цифрові перетворювачі (АЦП), датчики двійкової інформації, а до пристроїв виведення — індикатори, світлодіоди, дисплеї, друкувальні пристрої, цифроаналогові перетворювачі (ЦАП), транзисторні ключі, реле, комутатори. Пристрої введення-виведення відрізняються розрядністю даних, швидкодією, протоколами, тобто визначеним порядком обміну, керуючими сигналами. Дані у ПВВ змінюються у довільний або чітко визначений момент часу. З'єднання ПВВ з системною шиною МПС здійснюється за допомогою інтерфейсу введення-виведення, який узгоджує ПВВ з системною шиною МПС. Як правило, інтерфейс складається з одного або кількох *портів введення-виведення* та схем керування ними.

Під час проектування інтерфейсу введення-виведення треба забезпечити:

- зберігання інформації, яка надходить від ПВВ;
- доступ до інформації з боку МП;
- керування обміном;
- перетворення форматів даних.

Зберігання інформації і доступ до неї з боку МП. Введення і виведення інформації виконується за допомогою портів введення-виведення, які є 8- або 16-розрядними регістрами зі схемами вибірки та керування читанням-записом. Як порти можуть бути використані буферні регістри, наприклад *i*8282, *i*8285, КР580ИР82, КР589ИР12, КР580ВВ55. Використання регістра КР580ИР82 для з'єднання відповідно з

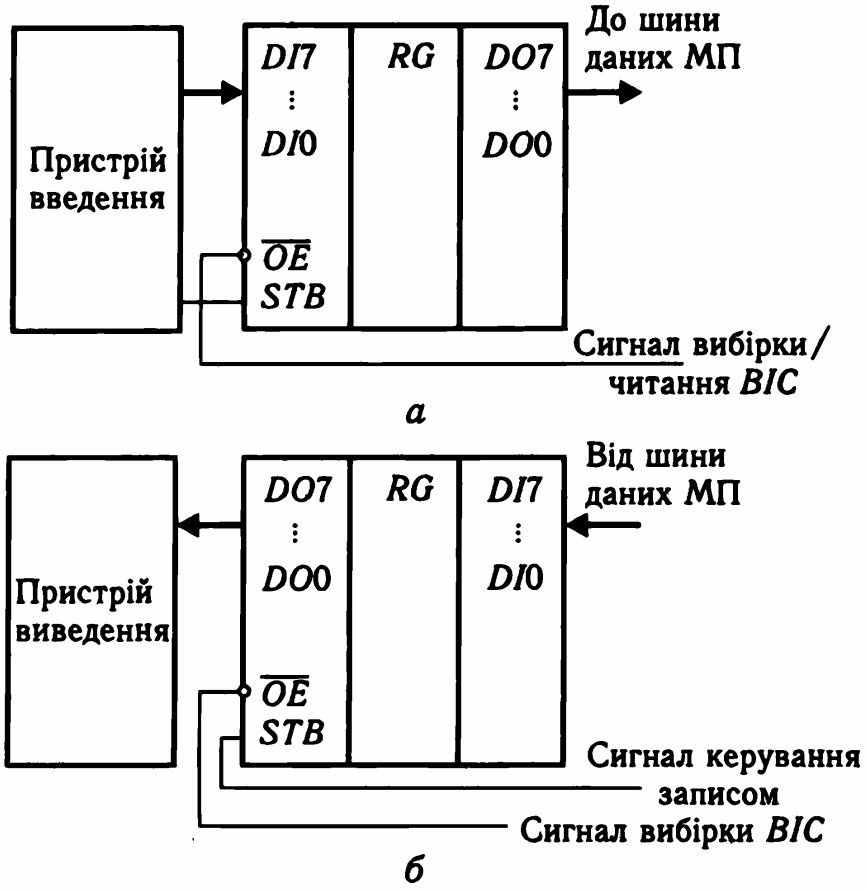


Рис. 5.1. Використання регістра КР580ИР82 для з'єднання: *а* — з пристроєм введення; *б* — з пристроєм виведення

пристроєм введення та пристроєм виведення показано на рис. 5.1, *а* і *б*.

Якщо регістр використовується як порт введення (рис. 5.1, *а*), то дані від пристрою введення надходять у регістр по лініях $DI7-DI0$ і записуються за стробом *STB*. Вихідні дані $DO7-DO0$ порту надходять у МПС по шині даних. МП формує також сигнал керування читанням і вибіркою порту, який надходить на вхід $\overline{OE}$. Якщо регістр використовується як порт виведення (див. рис. 5.1, *б*), то дані від МП надходять по шині даних на входи $DI7-DI0$ порту та супроводжуються сигналами керування записом і вибірки ВІС. Вихідні дані $DO7-DO0$ порту надходять у пристрій виведення.

Введення або виведення даних можна здійснювати двома способами:

- з використанням окремого адресного простору ПВВ;
- з використанням спільного з пам'яттю адресного простору, тобто з відображенням на пам'ять.

У першому випадку введення і виведення даних виконується за командами *IN* та *OUT* (див. табл. 2.11).

Приклад 5.1. Виконати виведення даних на 16-розрядний порт з адресою $1000H$.

Адреса порту займає два байти, тому для адресації порту слід використати непряму регістрову адресацію. Для цього треба адресу $1000H$ занести у регістр DX, а потім виконати команду виведення:

```
MOV DX, 1000H    ; Занести у DX число 1000H
OUT DX, AX       ; Вивести вміст AX на 16-розрядний порт виве-
                 ; дення з
                 ; адресою, яка знаходиться у DX, тобто AX →
                 ; → P16(DX).
```

Під час виконання команди $OUT\ DX,\ AX$ на лініях $A15-A0$ шини адреси з'являється адреса порту:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	= 1000H,

установлюється низький рівень сигналу запису введення-виведеня $\overline{IOW}$, і вміст акумулятора AX передається на 16-розрядну шину даних. Для фіксації даних (умісту AX) треба використати два 8-розрядних порти, один з яких з'єднаний з молодшою половиною шини даних, а другий — зі старшою. Сигнали керування записом і вибірки подаються паралельно на два порти.

Приклад 5.2. Виконати введення даних з 8-розрядного порту з адресою $32H$.

Введення даних здійснюється за командою введення:

$$IN\ AL,\ 32H;\quad AL \leftarrow P_8(32H).$$

Дії МП у цьому випадку аналогічні наведеним у прикладі 5.1. Відмінність полягає лише в тому, що активним стає сигнал $\overline{IOR}$ читання введення-виведення, і передавання інформації здійснюється від порту до МП по молодшій половині шини даних $D7-D0$.

За другим способом адреси портів розташовуються у спільному з пам'яттю адресному просторі, і звернення до портів не відрізняється від звернення до комірки пам'яті.

Сигнали вибірки ВІС (див. рис. 5.1, *б*) конкретних портів формуються за допомогою дешифраторів. Адреса 16-розрядного порту P_{16} має бути парною, щоб звернення до неї відбулося за один цикл шини. Адреси 8-розрядних портів введення-виведення P_8 можуть бути будь-якими (парними, непарними), але за парної адреси 8-розрядні порти слід з'єднати з молодшою половиною шини даних $D7-D0$, а за непарної — зі старшою половиною $D15-D8$.

Приклад 5.3. Розробити функціональну схему дешифратора (рис. 5.2) на ВІС К155ИД7 для адресації восьми 8-розрядних і восьми 16-розрядних портів, причому адреси 8-розрядних портів обрати непарними, а адреси 16-розрядних — парними.

Схема містить два ВІС дешифраторів $DC1$ і $DC2$. Із виходів дешифраторів сигнали надходять на входи $\overline{OE}$ відповідних портів. Тому ця схема (див. рис. 5.2) дає змогу адресувати 16 портів. Усі вихідні сигнали $0-7$ ВІС мають H-рівні, якщо не забезпечене надходження сигналів L-рівня на інверсні входи дозволу $E1$ і $E2$ та сигналу H-рівня на вхід $E3$. Інакше сигнал на виході DC, двійковий код номера якого визначається кодом на інформаційних входах DC $X2$, $X1$, $X0$, є актив-

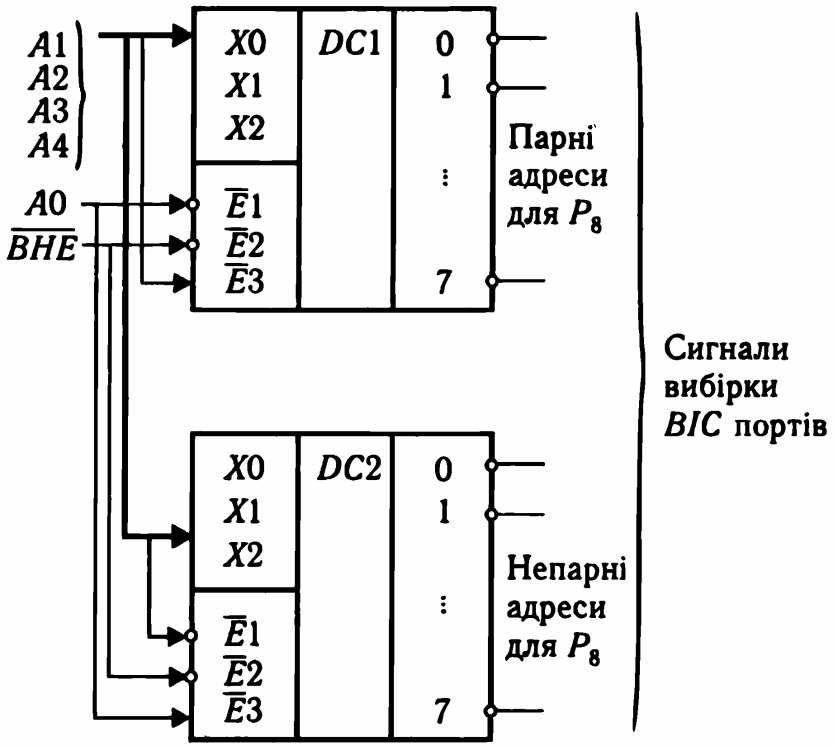


Рис. 5.2. Функціональна схема дешифратора портів

ним, тобто має L-рівень. Сигнали на інших виходах мають H-рівень. Цей принцип роботи DC дає змогу в будь-який момент роботи МП звернутися лише до одного з портів. Визначимо адреси портів.

Низькі рівні на виході $DC1$ з'являються, якщо значення сигналів на адресних лініях $A4 = 1$, $A0 = 0$, і сигнал вибірки старшого банку $\overline{BHE} = 0$. Сигнал на виході 0 $DC1$ буде активним (L-рівень) для адреси

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
x	x	x	x	x	x	x	x	x	x	x	1	0	0	0	0	= 0010H,

а на виході 1 — для адреси

x	x	x	x	x	x	x	x	x	x	x	1	0	0	1	0	= 0012H,

на виході 2 — для адреси

x	x	x	x	x	x	x	x	x	x	x	1	0	1	0	0	= 0014H,

.....

на виході 7 — для адреси

x	x	x	x	x	x	x	x	x	x	x	1	1	1	1	0	= 001EH.

Низькі рівні на виході $DC2$ з'являються якщо $A4 = 0$, $A0 = 1$, $\overline{BHE} = 0$. Сигнал на виході 0 $DC2$ буде активним (L-низький рівень) для адреси

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	1	= 0001H,

на виході 1 для адреси

x	x	x	x	x	x	x	x	x	x	x	0	0	0	1	1	= 0003H,

.....

на виході 7 для адреси

x x x x x x x x x x x 0 1 1 1 1 = 000*FH*.

Зазначимо, що 8-розрядні порти з непарними адресами мають бути з'єднаними зі старшою половиною шини даних *D*15–*D*8.

Керування обміном. Відомо три способи керування обміном:
- програмний обмін;
- обмін з перериванням;
- обмін у режимі прямого доступу до пам'яті.

Програмний обмін ініціюється МП і здійснюється під його керуванням. Розрізняють *простий програмний обмін* та *програмний обмін за стробом готовності*. За простого програмного обміну вважають, що ПВВ у будь-який момент готовий до обміну за командами *IN* або *OUT*. Під час обміну за стробом готовності ПВВ сповіщає про свою готовність до обміну стробом. Наприклад, видача 8-розрядних даних супроводжується дев'ятим бітом — стробом. У процесі такого обміну схема інтерфейсу містить тригер або порт керування для зберігання інформації про готовність зовнішнього пристрою до обміну. Процесор опитує відповідний розряд порту керування для визначення стану готовності зовнішнього пристрою.

Приклад 5.4. Розробити функціональну схему введення і виведення 8-розрядних даних за стробом готовності. Адреса порту введення — 02*H*, порту керування — 03*H*, порту виведення — 04*H*.

Функціональну схему обміну за стробом готовності зображено на рис. 5.3. Схема містить: пристрій введення, з'єднаний з портом введення; пристрій виведення, з'єднаний з портом виведення; порт керування для зберігання сигналів готовності пристроїв введення і виведення. Пристрій введення має вісім інформаційних вихідних ліній та одну вихідну лінію стробу супроводження даних. Поява цього стробу сигналізує про те, що дані на інформаційних лініях є дійсними (коректними). Пристрій виведення має вісім інформаційних вхідних ліній та одну вихідну лінію стробу підтвердження приймання даних. Поява цього стробу сигналізує про те, що дані прийняті пристроєм і МП може передавати нову порцію даних. Порт керування зберігає інформацію про строби від двох пристроїв.

Програма введення за стробом готовності має такий вигляд:

```
M1: IN AL, 03          ; AL ← порт керування (адреса 03)
    AND AL, 00000001B  ; Маскування всіх розрядів, крім D0
    JZ M1              ; Якщо D0 = 0 (порт не готовий), то на M1,
    IN AL, 02          ; інакше — введення інформації з порту
                       ; введення (адреса 02)
```

Програма виведення за стробом готовності:

```
M2: IN AL,03           ; AL ← порт керування (адреса 03)
    AND AL, 00000010B  ; Маскування всіх розрядів, крім D1
    JZ M2              ; Якщо D1=0 (порт не готовий), то на M2
    OUT 04, AL         ; інакше — виведення інформації на порт
                       ; виведення (адреса 04)
```

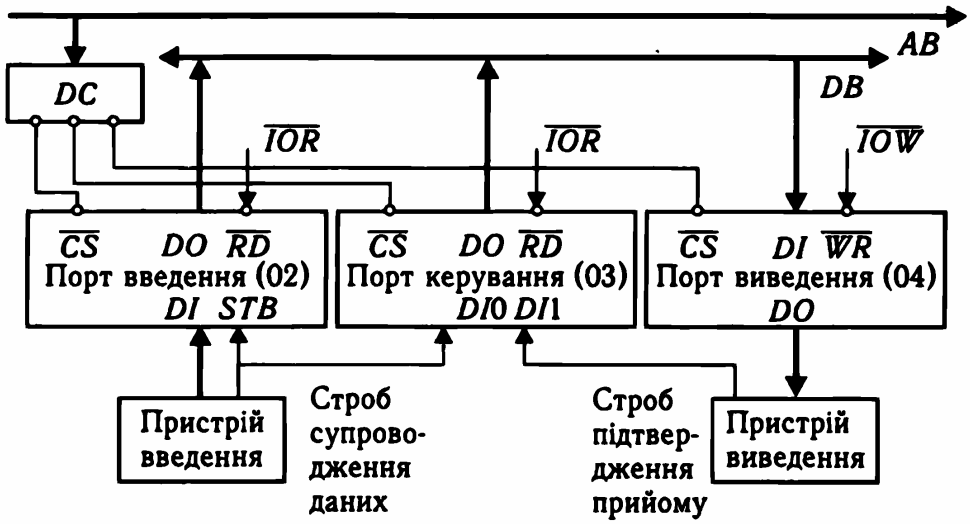


Рис. 5.3. Схема введення-виведення даних за стробом готовності

Якщо ПВВ має вбудований апаратний засіб для визначення готовності для обміну, то про стан пристрою свідчить прапорець готовності *READY* або прапорець готовності-зайнятості *READY/BUSY*. Інформація про готовність пристроїв належить до *статусної інформації* і входить до складу *слова стану* пристрою. Іноді стан готовності та зайнятості ідентифікується окремими прапорцями — *READY* і *BUSY*. Прапорець *READY* замінює біт порту керування (див. рис. 5.3).

Схему алгоритму програмного обміну даними за значенням прапорця *READY* наведено на рис. 5.4.

Якщо ПВВ не готовий до обміну, то МП знаходиться у режимі програмного очікування готовності зовнішнього пристрою, виконуючи команди блоків 1 і 2. Після виявлення стану готовності МП передає дані за командами блока 3, а потім працює з продовженням основної програми. На читання статусної інформації та її аналіз МП витрачає кілька циклів роботи, що призводить до непродуктивних втрат його часу. Недоліками програмного обміну за стробом готовності є те, що цей спосіб обміну інформацією не дає змоги зовнішнім пристроям ініціювати обмін. Перева-

Рис. 5.4. Алгоритм програмного введення

га програмного обміну полягає у простоті реалізації та у тому, що немає потреби у додаткових апаратних засобах.

Програмний обмін використовується для обміну з ПВВ, продуктивність яких менша, ніж продуктивність МП.

Обмін за перериванням ініціюється ПВВ і здійснюється під керуванням МП. У цьому разі сигнал готовності ПВВ до обміну використовується як запит переривання і надходить до програмованого контролера переривань (ПКП) (рис. 5.5). Введення або виведення здійснюється за підпрограмою оброблення запиту переривання.

Пристрій введення-виведення формує сигнал готовності *IRQ*, коли він готовий до обміну. Програмований контролер переривання (див. рис. 5.5) здатний сприйняти 8 сигналів *IRQ*7 — *IRQ*0. На рис. 5.5 сигнал готовності ПВВ надходить на вхід *IRQ*6. Сигнал готовності ПВВ — це вихідний сигнал тригера, який фіксує стан готовності *READY*. На виході програмованого контролера переривань асинхронно з діями МП формується сигнал *INT*. Заздалегідь не відомо, в який момент та які периферійні пристрої ініціюють переривання. Реагуючи на сигнал *INT*, МП перериває виконання програми, ідентифікує пристрій, переходить до підпрограми обслуговування переривань роботи цього пристрою, а після її завершення відновлює виконання перерваної програми. За командою *INT* уміст програмованого лічильника та прапорців автоматично запам'ятовується у стеку. Вміст акумулятора та РЗП треба занести у стек за допомогою команди *PUSH* у підпрограмі обробки переривання.

У кожному МП реалізовано особливу структуру системи переривань. Однак загальна послідовність обміну за перериванням містить такі дії:

- ПВВ генерує сигнал готовності, який викликає появу сигналу переривання, що подається на вхід *INT* МП;
- МП завершує виконання поточної команди і, якщо переривання дозволені (не замасковані), формує сигнал *INTA* підтвердження переривання;
- МП здійснює запам'ятовування вмісту акумулятора, програмованого лічильника, РЗП у стеку;
- МП ідентифікує пристрій, що зумовив переривання, і виконує відповідну підпрограму обслуговування переривання;
- за допомогою команди *POP* відновлюється значення вмісту акумулятора та РЗП зі стеку;
- за командою повернення з переривання *RET*, що є останньою командою підпрограми обслуговування переривання, відновлюються значення програмованого лічильника та прапорців і триває виконання перерваної програми.

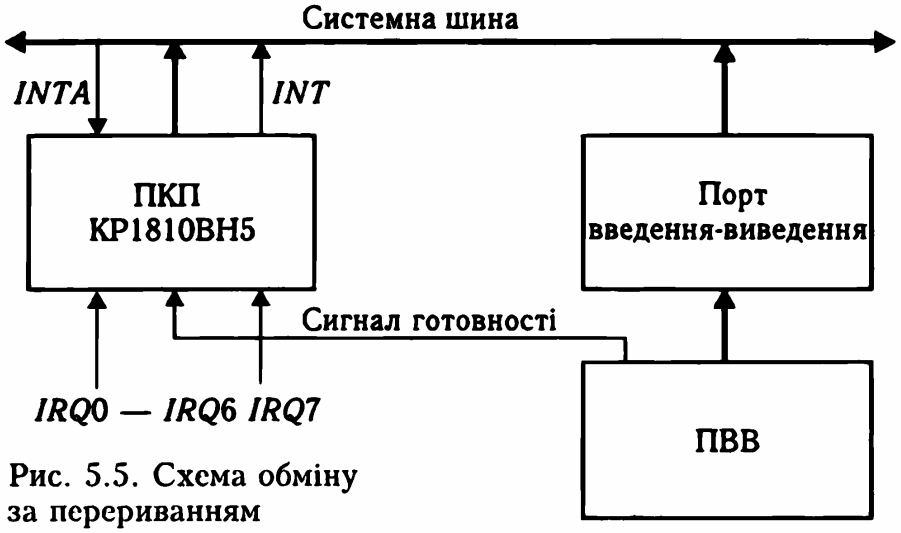


Рис. 5.5. Схема обміну за перериванням

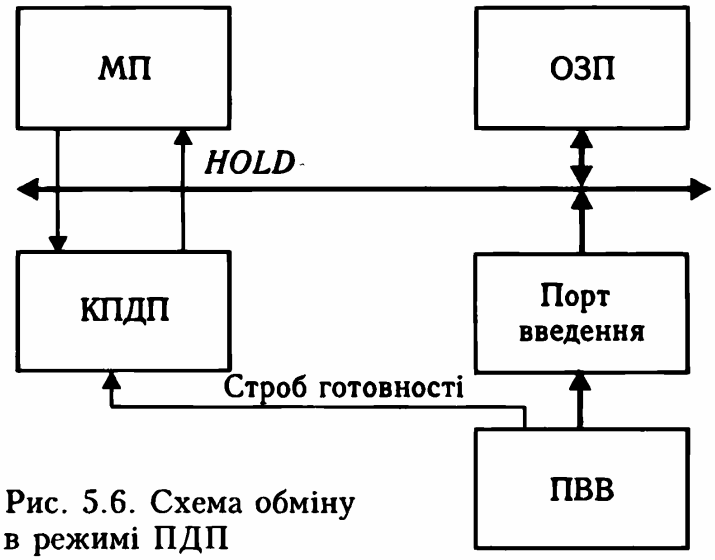


Рис. 5.6. Схема обміну в режимі ПДП

Обмін за перериванням продуктивніший, ніж програмний обмін, оскільки не потребує часу для опитування стану готовності ПВВ до обміну.

Обмін у режимі прямого доступу до пам'яті ініціюється ПВВ, здійснюється під керуванням контролера прямого доступу до пам'яті (КПДП) без участі МП. За необхідності обміну між ПВВ і пам'яттю немає потреби у пересиланні даних через МП. Дані за допомогою КПДП пересилаються безпосередньо з ПВВ у пам'ять або навпаки. Прямий доступ до пам'яті під час виконання операцій введення-виведення дає змогу значно збільшити швидкість передавання даних і підвищити ефективність використання засобів МП. Схему обміну в режимі ПДП зображено на рис. 5.6. Контролер прямого доступу приймає запит від ПВВ, формує сигнал запиту захоплення шин МП *HOLD* і, отримавши від МП дозвіл *HLDA*, формує адреси пам'яті та керуючі сигнали $\overline{MEMR}$, $\overline{IOW}$ — у разі

читання пам'яті, або $\overline{MEMW}$, $\overline{IOR}$ — у разі запису в пам'ять.

Інформацію про область пам'яті, що використовується для обміну у вигляді початкової адреси і довжини масиву, завантажують у КПДП під час програмування. Продуктивність обміну в режимі ПДП є найвищою.

Перетворення форматів даних. Якщо розрядність даних, з якими оперує МП, менша, ніж розрядність даних, з якими оперує ПВВ, то для узгодження розрядності збільшується кількість портів введення-виведення. Якщо розрядність даних, з якими оперує МП, більша, ніж розрядність даних, з якими оперує ПВВ, то для узгодження розрядності виконується пакування даних, отриманих з кількох джерел, в одне слово потрібної розрядності або доповнення нулями. Для перетворення послідовного коду на паралельний і навпаки використовують контролер послідовного обміну.

5.2. Програмований паралельний інтерфейс

Програмований паралельний інтерфейс КР580ВВ55 призначений для введення-виведення паралельної інформації у 8-байтовому форматі, що дає змогу реалізувати більшість відомих протоколів обміну по паралельних каналах. ВІС програмованого паралельного інтерфейсу можна використовувати для з'єднання МП зі стандартним периферійним устаткуванням — дисплеєм, телетайпом, накопичувачем тощо.

Структурну схему ВІС зображено на рис. 5.7, а її графічне позначення — на рис. 5.8. До складу ВІС входять (див. рис. 5.7):

- двоспрямований 8-розрядний буфер даних *BD (Bufer of Data)*, що з'єднує лінії даних ВІС із системною шиною даних;
- блок керування читанням-записом *RWCU (Read/Write Control Unit)*, що забезпечує керування зовнішнім і внутрішнім передаванням даних та керуючих слів;
- три 8-розрядних порти введення-виведення (*порт A, порт B, порт C*) для обміну інформацією, причому порт *C* поділений на два чотирирозрядних: C' ($PC7-PC4$) і C'' ($PC3-PC0$). Порти *A* і C' об'єднані у групу *B*, а порти *B* і C'' — у групу B'.

Схема ВІС містить також блоки керування групою *A* (*Control Unit A — CUA*) та групою *B* (*CUB*), що формують сигнали керування для відповідних груп.

Блок *RWCU* (*Register of Control Word Unit*) містить регістр керуючого слова, який зберігає керуючі слова, що надходять від МП.

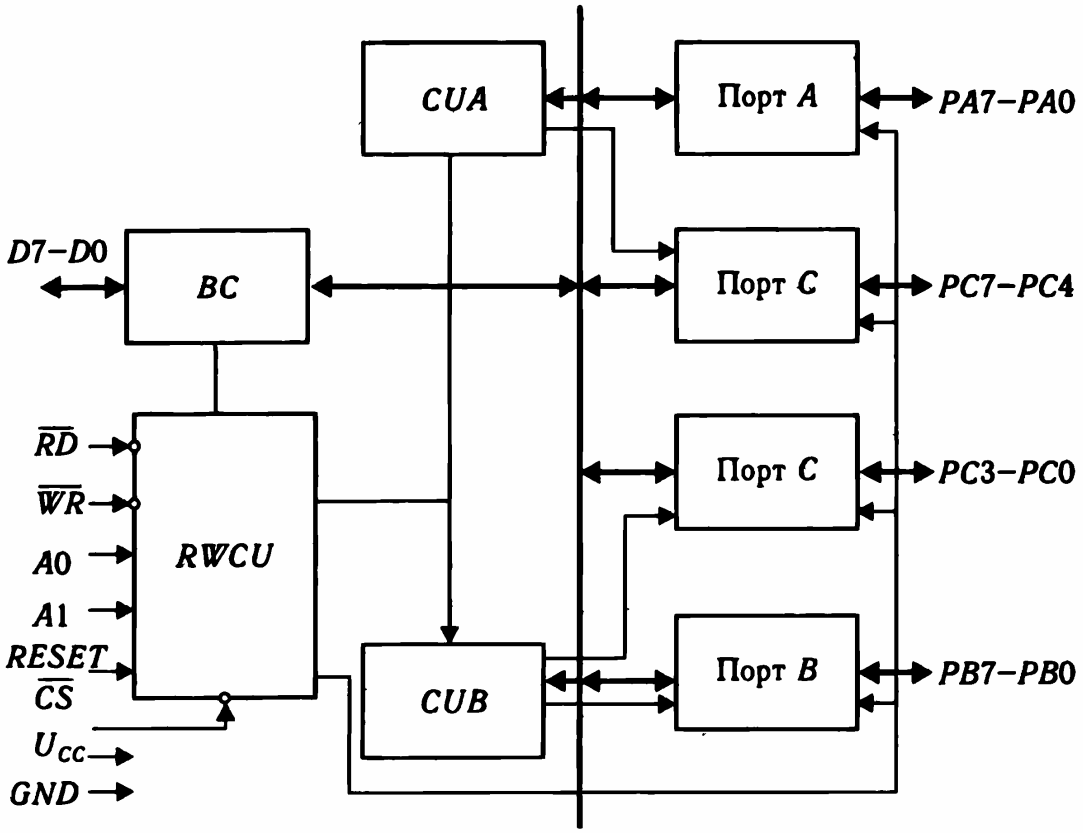
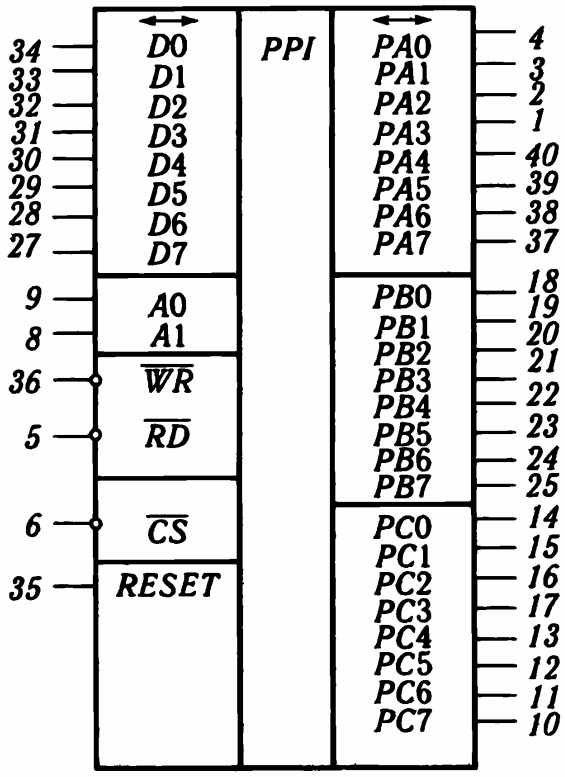


Рис. 5.7. Структурна схема ВІС КР580ВВ55



Таблиця 5.1. **Вибір портів ВІС КР580ВВ55**

A1	A0	Порт
0	0	A
0	1	B
1	0	C
1	1	RCW

Рис. 5.8. Графічне позначення ВІС КР580ВВ55

Таблиця 5.2. **Визначення виду операцій залежно від сигналів керування та адресних розрядів *A1*, *A0***

Операція	$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	*A1*	*A0*
Запис керуючого слова з МП	0	1	0	1	1
даних у порт *A*	0	1	0	0	0
даних у порт *B*	0	1	0	0	1
даних у порт *C*	0	1	0	1	0
Зчитування					
даних з порту *A*	0	0	1	0	0
даних з порту *B*	0	0	1	0	1
даних з порту *C*	0	0	1	1	0
Вимикання ВІС від *D7 – D0*	1	*x*	*x*	*x*	*x*

Примітка. *x* — будь-яке значення (0 або 1)

Таблиця 5.3. **Призначення виводів ВІС КР580ВВ55**

Позначення виводу	Номер виводу	Призначення виводів
D7-D0	27; 28; 29; 30; 31; 32; 33; 34	Вхід/вихід даних
$\overline{RD}$	5	Читання; *L*-рівень сигналу дозволяє зчитування інформації з регістра, що адресується розрядами *A0*, *A1* на лінії *D7 – D0*
$\overline{WR}$	36	Запис; *L*-рівень сигналу дозволяє запис інформації із шини *D7-D0* у порт паралельного інтерфейсу, що адресується розрядами *A0*, *A1*
A0, *A1*	9; 8	Входи для адресування внутрішніх регістрів ППУ
RESET	35	Скидання; *H*-рівень сигналу скидає регістр керуючого слова і встановлює всі порти у режим введення
$\overline{CS}$	6	Вхід вибірки мікросхеми; *L*-рівень сигналу з'єднує шину даних *D7 – D0* ВІС із системною шиною
PA7 – PA0	37; 38; 39; 40; 1; 2; 3; 4	Вхід-вихід порту *A*
PB7 – PB0	15; 24; 23; 22; 21; 20; 19; 18	Вхід-вихід порту *B*
PC7-PC0	10; 11; 12; 13; 17; 16; 15; 14	Вхід-вихід порту *C*
U_{CC}	26	Вивід напруги живлення +5 В
GND	7	Спільний вивід 0 В

Адресні розряди $A1$, $A0$ дають змогу обирати один з портів або регістр керуючого слова RCW (табл. 5.1).

Сигнал керування третім станом шини даних $\overline{CS}$, сигнали читання $\overline{RD}$, запису $\overline{WR}$ та скидання $RESET$ подаються на блок $RWCU$ і разом із сигналами на адресних лініях $A0$, $A1$ задають вид операції, що виконується (табл. 5.2).

Призначення виводів ВІС наведено у табл. 5.3.

Функціональну схему з'єднання ВІС із системною шиною МП зображено на рис. 5.9.

Відповідно до схеми рис. 5.9 і табл. 5.2 визначаються адреси портів та регістра керуючого слова RCW (табл. 5.4).

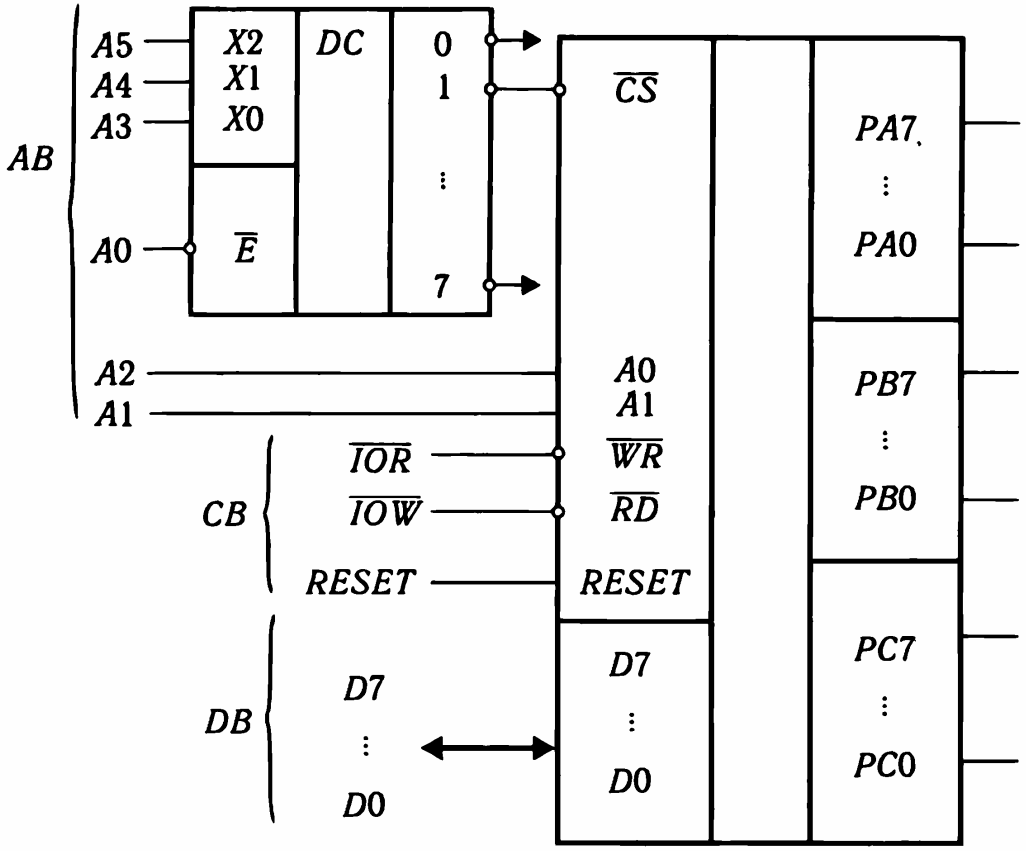


Рис. 5.9. Функціональна схема з'єднання ВІС КР580ВВ55 із системною шиною

Таблиця 5.4. **Адреси портів і регістра *RCW***

Порт	*A7*	*A6*	*A5*	*A4*	*A3*	*A2*	*A1*	*A0*	Адреса
A	0	0	0	0	1	0	0	0	08*H*
B	0	0	0	0	1	0	1	0	0A*H*
C	0	0	0	0	1	1	0	0	0C*H*
RCW	0	0	0	0	1	1	1	0	0E*H*

D7	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*
1	*M1*	*M0*	*IOA*	*IOC'*	*M*	*IOB*	*IOC''*

Рис. 5.10. Формат керуючого слова режиму

Програмування BIC полягає у завантаженні керуючого слова режиму за $A1 = 1$, $A0 = 1$. Формат керуючого слова режиму показано на рис. 5.10. Керуюче слово визначає один з трьох режимів портів паралельного інтерфейсу: режим 0 — основний режим введення-виведення; режим 1 — режим введення/виведення за стробом готовності; режим 2 — режим двонапрямленого передавання інформації.

На рис. 5.10 прийнято таке позначення:

M1, *M0* — біти, що задають режим групи *A*. За $M1M0 = 00$ формується режим 0, за 01 — режим 1, за $1x$ — режим 2;

IOA — біт, що задає режим введення або виведення порту *A*. За $IOA = 1$ здійснюється введення інформації, за 0 — виведення;

IOC' — біт, що задає режим введення або виведення порту *C'* (1 — введення, 0 — виведення);

M — біт, що задає режим групи B. За $M = 0$ — режим 0, 1 — режим 1;

IOB — біт, що задає режим введення або виведення порту *B* (1 — введення інформації, 0 — виведення);

IOC'' — біт, що задає режим введення або виведення порту *C''* (1 — введення, 0 — виведення).

Керуюче слово може встановлювати різні режими роботи для кожного з портів. Порт *A* може працювати у будь-якому з трьох режимів, порт *B* — у режимах 0 та 1. Порт *C* можна використовувати для передавання даних лише у режимі 0; в інших режимах його застосовують для передавання керуючих сигналів, що супроводжують процес обміну в портах *A* і *B*.

Окремі розряди порту *C* можна встановлювати або скидати програмно за допомогою керуючого слова встановлення-скидання, формат якого показано на рис. 5.11.

На рис. 5.11 позначено:

біти *N2*, *N1*, *N0* — задають номер розряду, який треба встановити або скинути. Значення цих бітів: 000 — визначає

D7	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*
0	*x*	*x*	*x*	*N2*	*N1*	*N0*	*S/R*

Рис. 5.11. Формат керуючого слова встановлення-скидання порту *C*

розряд *PC*0, 001 — *PC*1, 010 — *PC*2, 011 — *PC*3, 100 — *PC*4, 101 — *PC*5, 110 — *PC*6, 111 — *PC*7;

біт *S*/*R* — задає режим установлення або скидання розряду порту *C*, який обрано значеннями *N*2, *N*1, *N*0. За *S*/*R* = = 1 відбувається встановлення розряду, за 0 — скидання.

Приклад 5.5. Сформувати імпульс тривалістю *n* (мкс). Адреси ВІС КР580ВВ55 подано в табл. 5.4.

Для того щоб сформувати імпульс заданої тривалості, треба встановити розряд *PC*4 у стан логічної одиниці, потім виконати підпрограму затримки на *n* (мкс) й провести скидання розряду *PC*4.

Визначимо керуючі слова для встановлення і скидання розряду *PC*4. Керуюче слово встановлення розряду *PC*4 згідно з рис. 5.11 буде таким:

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
0	0	0	0	1	0	0	1

У 16-річній системі числення воно дорівнює 09*H*.

Керуюче слово скидання розряду *PC*4:

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
0	0	0	0	1	0	0	0

У 16-річній системі числення воно дорівнює 08*H*.

Програма формування імпульсу тривалістю *n* (мкс) на виході *PC*4 порту *C*:

```
MOV     AL,09          ; Формування керуючого слова, встановлен-
                       ; ня розряду PC4
OUT     0EH,AL         ; Запис умісту AL до регістра керуючого
                       ; слова
CALL    DELAY          ; Затримка часу на n (мкс)
MOV     AL,08          ; Формування керуючого слова скидання
                       ; розряду PC4
OUT     0EH,AL         ; Виведення до регістра керуючого слова
        ..........................................................
                       ; Підпрограма затримки на n (мкс)
DELAY:  MOV CX,134     ; CX←134 (4 такти)
D:      LOOP D         ; CX←CX — 1, якщо не 0, то перехід на D
                       ; під час
                       ; виконання переходу на мітку D команда
                       ; виконується
                       ; за16 тактів, якщо CX = 0 — за 4 такти
        RET            ; Повернення з підпрограми (8 тактів)
```

Визначимо тривалість затримки у цьому прикладі. У коментарі до команд підпрограми затримки у дужках запишемо тривалість виконання команд у тактах (див. табл. 2.11). Загальна кількість тактів, потрібна для виконання підпрограми *DELAY*, становить:

$$4 + 134 \times 16 + 4 + 8 = 2160.$$

За тактової частоти 5 МГц тривалість одного такту дорівнює 200 нс. Тоді значення n визначається як

$$n = 0{,}2 \times 2159 = 431{,}8 \text{ мкс}.$$

Для кожної групи A або B у ВІС є тригер дозволу переривання $INTE$, встановлення-скидання якого здійснюється керуючим словом встановлення-скидання визначеного розряду порту C. Якщо тригер дозволу переривання відповідної групи встановлений ($INTE = 1$), то паралельний інтерфейс може сформувати запит переривання за готовності ПВВ до обміну.

Розглянемо режими роботи портів програмованого паралельного інтерфейсу.

Режим 0 застосовується за синхронного обміну або під час програмної організації асинхронного обміну. В цьому режимі ВІС — це пристрій, що складається з чотирьох портів (два 8-розрядних і два 4-розрядних), які можуть незалежно налагоджуватися на введення або виведення інформації. Виведення інформації здійснюється за командою OUT з фіксацією виведеної інформації у регістрах портів, а введення — за командою IN без запам'ятовування інформації.

Приклад 5.6. Установити порт A у режим введення 0, порт B — у режим виведення 0, порт C' — у режим введення 0, порт C'' — у режим виведення 0, а потім здійснити відповідно введення або виведення інформації через порти A і B.

Керуюче слово режиму в цьому разі таке:

$D7$	$D6$	$D5$	$D4$	$D3$	$D2$	$D1$	$D0$
1	$M1$	$M0$	IOA	IOC'	M	IOB	IOC''
1	0	0	1	1	0	0	0

$= 98H.$

Програма матиме такий вигляд:

```
MOV   AL, 98H     ; Формування керуючого слова режиму 98H у AL
OUT   OEH,AL      ; Запис до регістра RWC
⋮
IN    AL,08H      ; Введення з порту A
⋮
OUT   0AH, AL     ; Виведення на порт B
```

Зазначимо, що в цьому прикладі адреси портів визначаються за схемою рис. 5.9.

Режим 1 забезпечує односпрямований обмін інформацією МП з ПВВ за стробом готовності. Інформація передається по портах A і B, а лінії порту C керують процесом передавання. Роботу порту в режимі 1 супроводжують три керуючі сигнали. Якщо один з портів запрограмовано на режим 1, то інші 13 інтерфейсних ліній можна використовувати у режимі 0. Якщо обидва порти запрограмовано на режим 1, то дві інтерфейсні лінії порту C, що залишилися, можуть бути налагоджені на введення або виведення. Призначення розрядів порту C під час введення даних з портів A і B у режимі 1 показано на рис. 5.12.

Приклад схеми з'єднання пристрою введення з портом A, пристрою виведення — з портом B у режимі 1 показано на рис. 5.13.

Група *A*				Група *B*			
PC7	*PC6*	*PC5*	*PC4*	*PC3*	*PC2*	*PC1*	*PC0*
IO	*IO*	*IBF A*	*INTE A*	*INTR A*	*INTE B*	*IBF B*	*INTR B*

Рис. 5.12. Призначення розрядів порту *C* під час введення даних з портів *A* і *B* у режимі 1:
IBF (Input Buffer Full) — вихідний сигнал ВІС, який повідомляє про заповненість вхідного буфера порту даними; *INTR (INTeRrupt)* — вихідний сигнал, що повідомляє про завершення приймання інформації; *INTE (INTerrupt Enable)* — сигнал дозволу переривання (вхід стробу приймання)

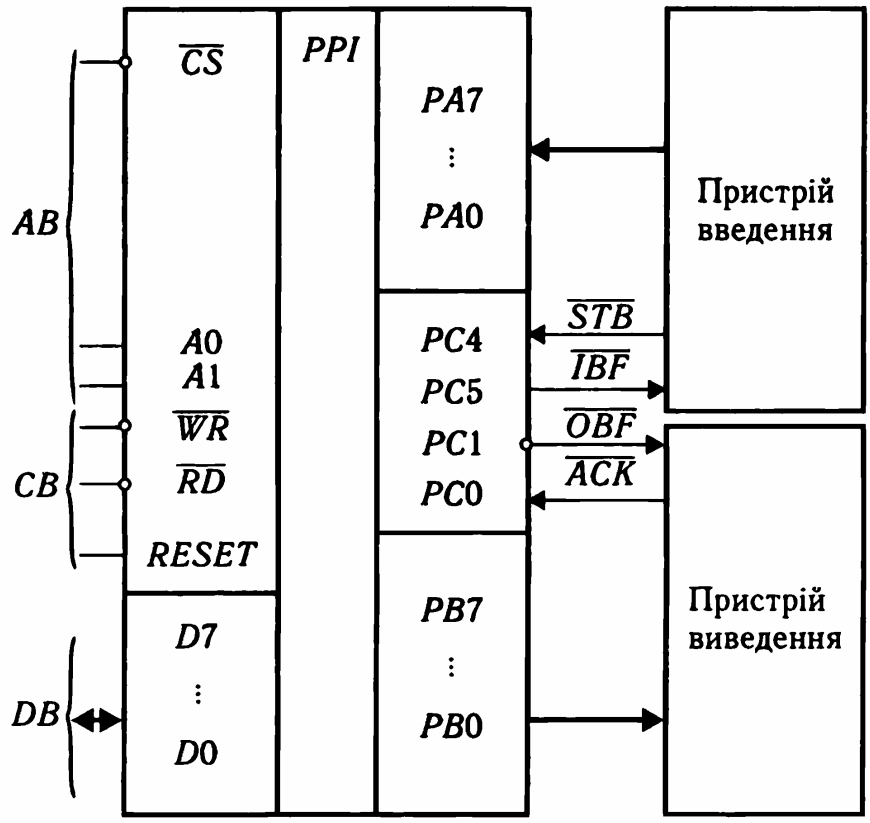


Рис. 5.13. Схема з'єднання пристрою введення з портом *A*, пристрою виведення — з портом *B* у режимі 1

Введення даних у режимі 1 здійснюється по каналу *A*, а керуючі сигнали передаються по лініях *PC4* і *PC5*. Пристрій введення видає строб приймання $\overline{STB}$, який вказує на готовність до введення інформації. Цей строб надходить на вхід дозволу переривання від каналу *A* — *PC4*. Вихідний сигнал *IBF* з виводу *PC5* використовується для підтвердження приймання. Він формується за спадом *STB* і повідомляє пристрій введення про закінчення приймання даних.

Крім показаних на рис. 5.13 сигналів, програмований паралельний інтерфейс формує також сигнал запиту переривання *INTR*, який інфор-

Група *A*				Група *B*			
PC7	*PC6*	*PC5*	*PC4*	*PC3*	*PC2*	*PC1*	*PC0*
$\overline{OBF}$ *A*	*INTE A*	*IO*	*IO*	*INTR A*	*INTE B*	$\overline{OBF}$ *B*	*INTR B*

Рис. 5.14. Призначення розрядів порту *C* під час виведення даних на порти *A* і *B* в режимі 1

мує МП про завершення приймання інформації. Під час обміну за перериванням цей сигнал використовується як запит переривання, а за програмного обміну може ігноруватися. Високий рівень цього сигналу встановлюється, якщо $\overline{STB} = 1$, $IBF = 1$. Нульовий рівень сигналу *INTR* установлюється з надходженням сигналу $\overline{IOR}$.

Призначення розрядів порту *C* у режимі 1 виведення показано на рис. 5.14. Сигнал $\overline{OBF}$ (*Output Buffer Full*) — вихідний сигнал, що повідомляє про заповненість вихідного буфера порту даними.

Інші сигнали мають таке саме значення, що й на рис. 5.12.

Приклад виведення даних на порт *B* у режимі 1 показано на рис. 5.13. Для виведення даних у цьому режимі використовують такі керуючі сигнали: $\overline{OBF}$ — вихідний сигнал, що формується за фронтом $\overline{WR}$ та повідомляє ПВВ про готовність до виведення; *ACK* — вхідний сигнал, що підтверджує приймання інформації з ВІС інтерфейсу; *INTR* — вихідний сигнал ВІС, що повідомляє МП про завершення виведення. Сигнал *INTR* встановлюється у стан логічної одиниці за $\overline{OBF} = 1$ і $\overline{ACK} = 1$, і відбувається скидання у стан логічного нуля сигналом $\overline{IOW}$ у процесі запису даних у паралельний інтерфейс.

Розряди *PC6*, *PC7* (див. рис. 5.12) та *PC5*, *PC4* (див. рис. 5.14) під час виведення не беруть участі у керуванні обміном і можуть бути запрограмовані на просте введення або виведення (*I/O*). Введення здійснюється читанням порту *C*, а виведення — записом керуючих слів встановлення-скидання окремих розрядів (див. рис. 5.11).

Обмін за стробом готовності може здійснюватися за перериванням або за програмою. Під час обміну за перериванням сигнал *INTR* надходить до системи переривання та ініціює обмін. Під час обміну за програмою готовність портів *A* або *B* визначається опитуванням *INTR A* або *B*.

Приклад 5.7. Написати програму встановлення порту *A* у режим уведення 1 та ввести дані за стробом готовності. Адреси порту *A* визначають за допомогою рис. 5.9.

Керуюче слово режиму в цьому випадку:

D7	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*	
1	*M1*	*M0*	*IOA*	*IOC'*	*M*	*IOB*	*IOC''*	
1	0	1	1	0	0	0	0	= 0B0*H*.

Програма має такий вигляд:

```
MOV   AL, 0B0H          ; Формування керуючого слова режи-
                        ; му в AL
OUT   0EH,AL            ;Запис до регістра RWC ВІС КР580ВВ55
MOV   AL,09             ; формування керуючого слова вста-
                        ; новлення розряду
                        ; PC4 — INTE A — дозвіл переривань
OUT   0EH,AL            ; Запис вмісту AL до регістра керуючо-
                        ; го слова
      ⋮
M1:   IN AL,0CH         ; AL ← вміст порту C
      AND AL, 00001000B ; Маскування всіх розрядів, крім PC3
                        ; (INTR A)
      JZ M1             ; Якщо дані не готові, то на M1,
      IN AL, 08H        ; інакше — введення інформації з пор-
                        ; ту A
      ⋮
```

Режим 2 забезпечує двонапрямлене передавання інформації з порту *A* до зовнішнього пристрою і навпаки. Процес обміну супроводжують 5 керуючих сигналів, що подаються по лініях *PC7*—*PC3* (див. рис. 5.14); 11 інтерфейсних ліній, що залишилися, можуть налагоджуватися на режим 0 або режим 1. Призначення розрядів порту *C* у режимі 2 наведено на рис. 5.15, а схема з'єднання програмованим паралельним інтерфейсом — на рис. 5.16.

PC7	*PC6*	*PC5*	*PC4*	*PC3*	*PC2*	*PC1*	*PC0*
OBF A	*INTE1*	*IBF A*	*INTE2*	*INTR A*	*x*	*x*	*x*
					Залежать від режиму порту *B*		

Рис. 5.15. Призначення розрядів порту *C* у режимі 2

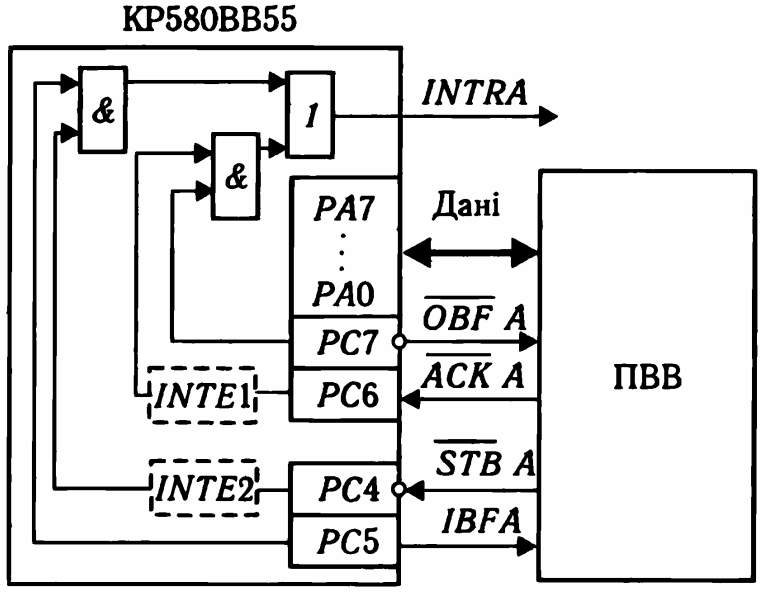


Рис. 5.16. Схема під'єднання ПВВ до ВІС КР580ВВ55 у режимі 2

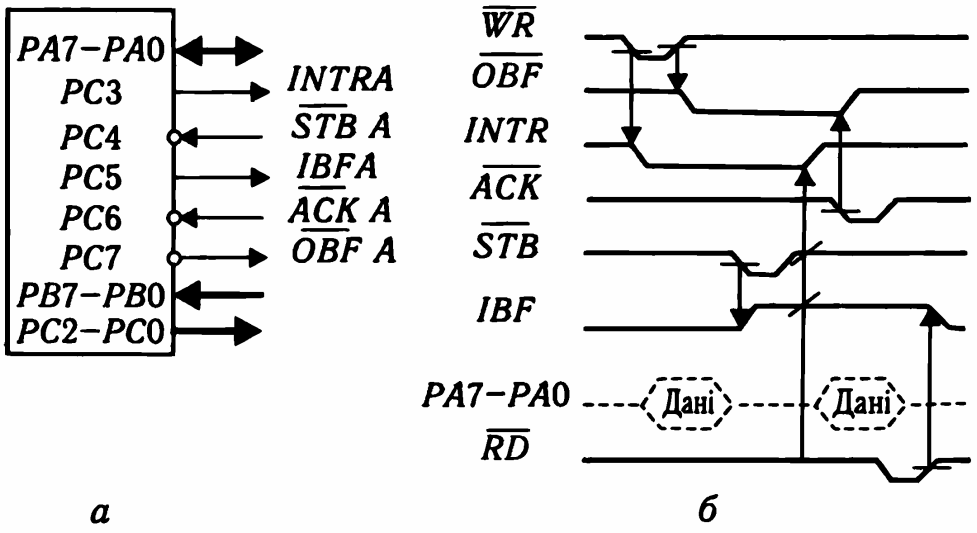


Рис. 5.17. Робота ВІС у режимі 2:
а — розподіл сигналів по інтерфейсних лініях; *б* — часові діаграми роботи

Призначення керуючих сигналів у режимі 2 аналогічне призначенню сигналів у режимі 1. Керування формуванням внутрішнього сигналу *INTE* для операції введення здійснюється по лінії *PC*4, а для операції виведення — по лінії *PC*6.

Вивід ВІС *INTR A* використовують як запит переривань під час введення, так і під час виведення інформації. Розподіл сигналів по інтерфейсних лініях, керуюче слово у режимі 2 та часові діаграми роботи подано на рис 5.17.

Приклад 5.8. Написати програму встановлення порту *A* у режим введення 2, а потім здійснити введення (виведення) інформації через порт *A* в цьому самому режимі.

Керуюче слово режиму в цьому випадку дорівнюватиме 0*C*0*H*:

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
1	*M*1	*M*0	*IOA*	*IOC'*	*M*	*IOB*	*IOC''*
1	1	0	0	0	0	0	0

Програма двонапрямленого введення-виведення за стробом готовності має спочатку виявити готовність порту до введення або виведення за одиничним станом сигналу *INTR A* (лінія *PC*3), а потім встановити, які саме дані готові — для введення (одиничний стан лінії *PC*4) чи виведення (одиничний стан лінії *PC*6). Після цього можна здійснювати обмін даними. Програма має такий вигляд:

```
        MOV AL, 0C0H        ; Формування керуючого слова режиму в AL
        OUT 0EH, AL         ; Запис до регістра RWC ВІС КР580ВВ55
        ⋮
M1:     IN AL, 0CH          ; AL ← вміст порту C
        MOV BL, AL          ; Зберігання вмісту AL у регістрі BL
        AND AL, 00001000B   ; Маскування всіх розрядів, крім PC3
                            ; (INTR A)
        JZ M1               ; Якщо дані не готові, то на M1
```

```
        MOV AL,BL          ; AL ← вміст порту C
        AND AL, 00010000B  ; Маскування всіх розрядів, крім PC4
                           ; (INTR A)
        JZ M2              ; Якщо дані для введення не готові, то пе-
                           ; рехід на
        IN AL, 08H         ; виведення інакше — введення інформації
                           ; з порту A
        JMP M3
M2:     OUT 08H, AL        ; Виведення інформації на порт A
M3:                        ; Продовження програми
```

5.3. Програмований інтерфейс клавіатури та індикації

Програмований інтерфейс клавіатури й індикації КР580ВВ79 призначений для реалізації обміну інформацією між МП і матрицею клавіш (датчиків) та індикації. На відміну від ВІС паралельного інтерфейсу КР580ВВ55 (див. п. 5.2), який може використовуватися для будь-якого пристрою, що здійснює введення-виведення даних у паралельному форматі, програмований інтерфейс клавіатури та індикації є спеціалізованим і призначений для обміну інформацією лише з деякими типами клавіатури й індикаторів. Структурну схему ВІС зображено на рис. 5.18. Позначення виводів подано в табл. 5.5.

Схема містить двонапрямлений 8-розрядний буфер даних *BD*, що з'єднує лінії даних ВІС із системною шиною даних; блок *RWCU*, що забезпечує керування зовнішнім і внутрішнім передаванням даних та керуючих слів; блок керування; блок інтерфейсу індикації; блок інтерфейсу клавіатури.

Блок керування містить схему керування та синхронізації і лічильник сканування *CT*. Схема керування і сигналізації формує сигнали, які керують усіма блоками ВІС, сигнали внутрішньої синхронізації та сигнал $\overline{BD}$ для погашення індикатора під час зміни символів. До складу схеми входить подільник частоти з програмованим коефіцієнтом ділення для генерації внутрішніх імпульсів синхронізації частотою до 100 кГц. Лічильник сканування формує коди на лініях *S*3—*S*0 для опитування матриці клавіатури та керування індикацією. При цьому залежно від керуючих слів можна налагоджувати схеми видавання стану лічильника сканування або на безпосереднє виведення вмісту чотирьох молодших розрядів лічильника або на виведення вмісту двох молодших розрядів через дешифратор з чотирма виходами.

Блок інтерфейсу індикації містить ОЗП індикації *RAM* 1 інформаційною ємністю 16 × 8, адресний регістр

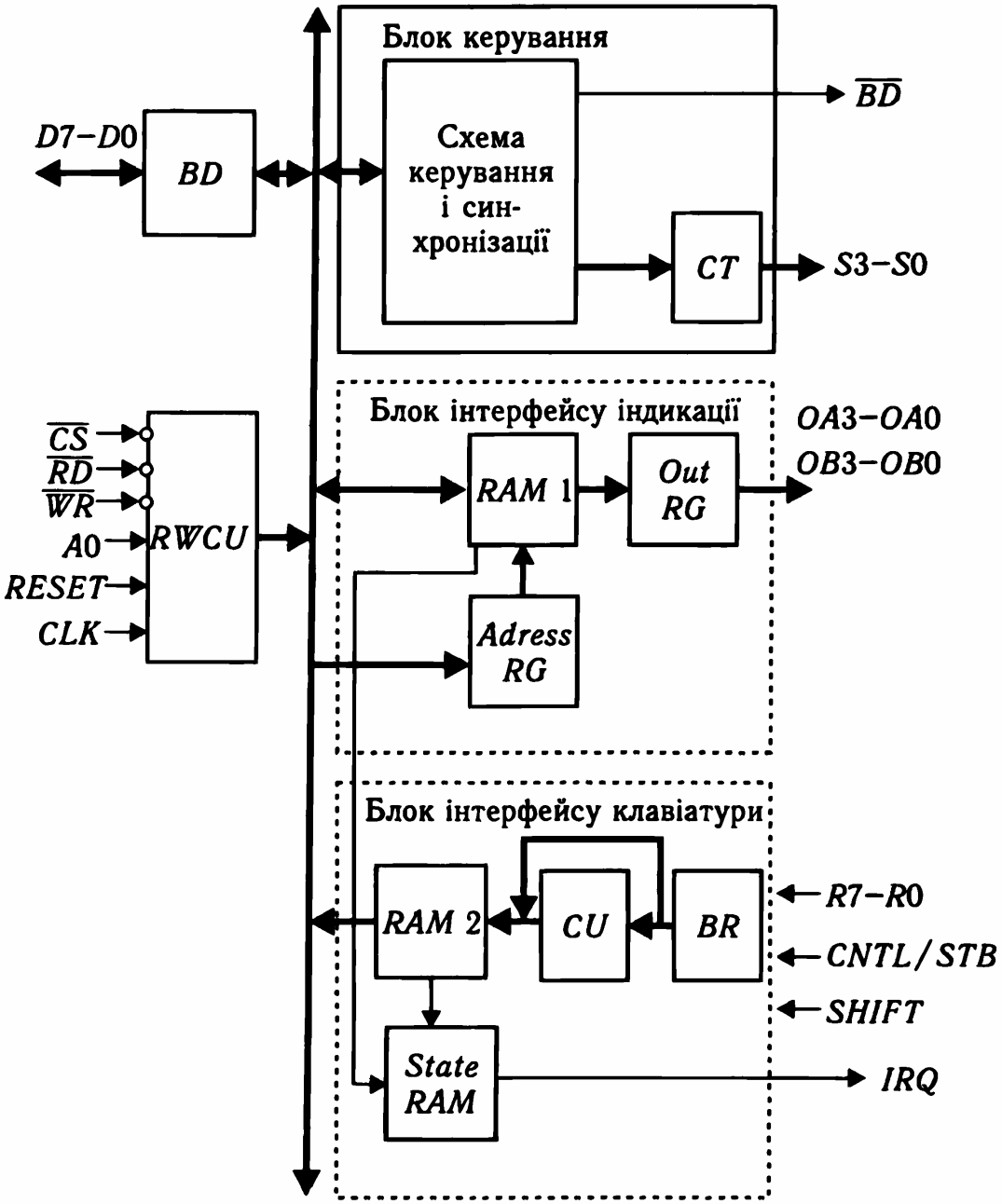


Рис. 5.18. Структурна схема програмованого інтерфейсу клавіатури індикації

Address RG та вихідний регістр *Out RG*. ОЗП складається з двох незалежних частин по 16 чотирирозрядних слів кожний та зберігає коди символів для індикації на 8- або 16-знакомісць. Тип індикації задається відповідним керуючим словом. Дані з ОЗП передаються через вихідні регістри на виходи *OA3—OA0* (старша частина 8-розрядного слова) та *OB3—OB0* (молодша частина).

Блок інтерфейсу клавіатури має ОЗП *RAM2* клавіатури-датчиків, буфер повернення *BR*, схему усунення брязкоту контактів *CU*, схему аналізу стану ОЗП *State RAM*.

Таблиця 5.5. **Призначення виводів ВІС КР580ВВ79**

Позначення виводу	Номер виводу	Призначення виводів
*D*0 – *D*7	12; 13; 14; 15; 16; 17; 18; 19	Вхід-вихід даних
CLK	3	Вхід синхросигналів. Частота не повинна перевищувати 3,2 МГц
$\overline{RD}$	10	Читання; *L*-рівень сигналу дозволяє зчитування інформації з регістра, що адресується розрядами *A*0, *A*1 на лінії *D*7 – *D*0
$\overline{WR}$	11	Запис; *L*-рівень сигналу дозволяє запис інформації з шини *D*7 – *D*0 у порт ВІС, що адресується розрядами *A*0, *A*1
*A*0	21	Якщо *A*0 = 1, у ВІС передається керуюче слово або з нього зчитується слово стану, а якщо *A*0 = 0, передаються дані
RESET	9	Скидання; *H*-рівень сигналу скидає у початковий стан
$\overline{CS}$	22	Вхід вибірки мікросхеми; *L*-рівень сигналу з'єднує шину даних *D*7 – *D*0 ВІС із системною шиною
*S*0 – *S*3	32; 33; 34; 35	Лінії сканування. Як клавіатури (матриці датчиків), так і позицій дисплею можуть працювати у режимі лічильника або інверсного дешифратора
*R*0 – *R*7	38; 39; 1; 2; 5; 6; 7; 8	Лінії повернення. З'єднуються з лініями сканування через клавіатуру (матрицю датчиків). Натискання клавіші приводить до появи нуля на одній з ліній повернення
SHIFT	36	Зсув. Стан цієї лінії запам'ятовується у коді клавіші і може бути використаний як ознака перемикання клавіатури
CNTR/STB	37	У режимі клавіатури використовується так само, як і лінія *SHIFT*. У режимі введення за стробом використовується як вхід стробу (введення здійснюється за переднім фронтом *STB*)
U_{CC}	26	Вивід напруги живлення +5 В
GND	7	Спільний вивід 0 В
*OA*0 – *OA*3 *OB*0 – *OB*3	27; 26; 25; 24 31; 30; 29; 28	Виходи регістрів даних дисплею. Можуть бути використані як один 8-розрядний або два 4-розрядні виходи
$\overline{BD}$	23	Гасіння дисплея під час перемикання цифр або під час видання керуючого слова очищення дисплея
U_{CC}	40	Вивід напруги живлення +5 В
GND	20	Спільний вивід 0 В

Він забезпечує введення інформації через лінії повернення ($R7-R0$) з клавіатури. Збереження введеної інформації здійснюється в ОЗП *RAM2*, який є стеком ємністю 8×8 біт. Вхідні лінії $R7-R0$ мають високий внутрішній опір, що дає змогу безпосередньо під'єднувати до них матриці клавіатури або датчиків. Для забезпечення режиму введення даних з датчиків за стробом готовності передбачено лінію *CNTL/STB*.

Виходи буфера *BR* з'єднані з входами схеми усунення брязкоту контактів, яка виявляє заборонені ситуації під час натискання клавіш і не допускає повторного введення коду клавіш, що може статися внаслідок брязкоту контактів. Схема аналізу стану ОЗП формує статусну інформацію про роботу ОЗП та сигнал запиту переривання *IRQ*.

Функціональну схему з'єднання ВІС із системною шиною МП зображено на рис. 5.19.

Програмування мікросхеми КР580ВВ79 здійснюється завантаженням *керуючого слова ініціалізації клавіатури* і *дисплея* у відповідний регістр керуючих слів, розташованого у блоці керування. Під час записування керуючих слів на

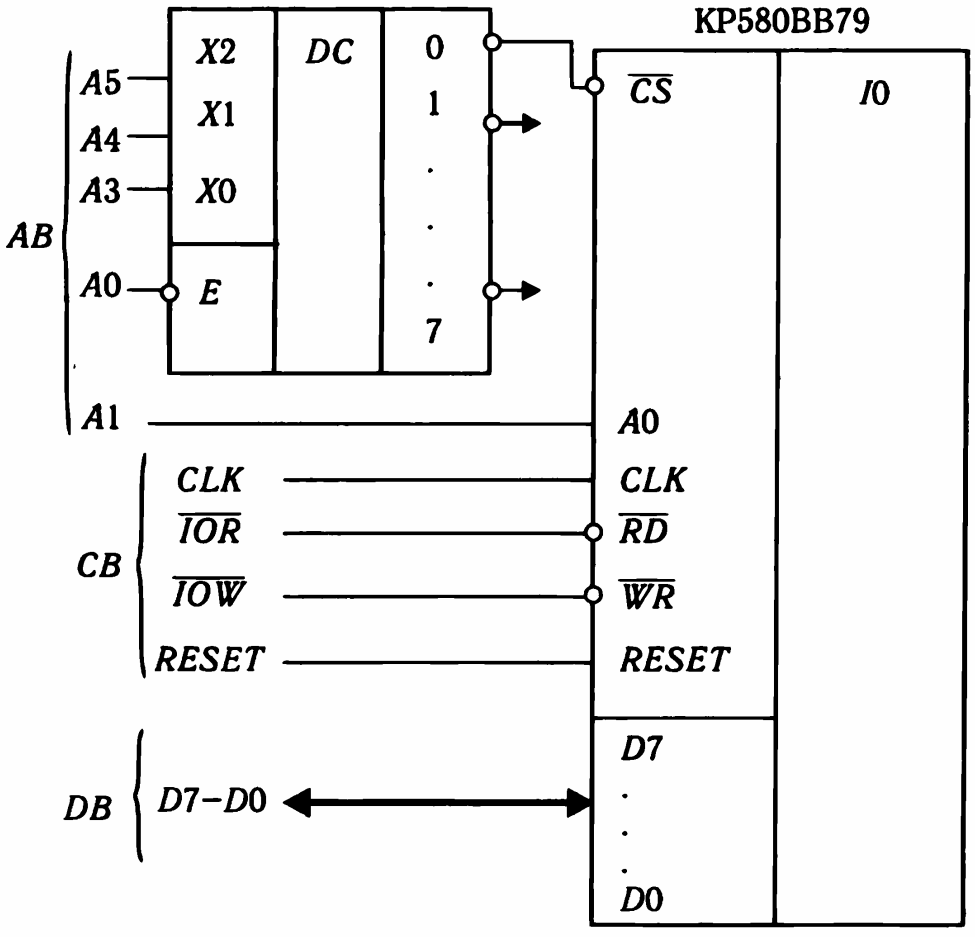


Рис. 5.19. З'єднання ВІС КР580ВВ79 із системною шиною МПС

Рис. 5.20. Формат керуючого слова ініціалізації клавіатури і дисплея

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
0	0	0	*D*	*D*	*K*	*K*	*S*

вхід *A*0 слід подавати сигнал логічної одиниці. Формат керуючого слова ініціалізації клавіатури і дисплея показано на рис. 5.20.

Розряди *D*3 і *D*4 керуючого слова (рис. 5.20) визначають режим роботи дисплея:

Розряд		Режим роботи дисплея
*D*4	*D*3	
0	0	Дисплей на 8 символів із введенням з лівого боку
0	1	Дисплей на 16 символів із введенням з лівого боку
1	0	Дисплей на 8 символів із введенням з правого боку
1	1	Дисплей на 16 символів із введенням з правого боку

Розряди *D*1 та *D*2 визначають режим роботи клавіатури:

Розряд		Режим роботи дисплея
*D*1	*D*2	
0	0	Клавіатура у режимі одиночного натискання клавіш
0	1	Клавіатура у режимі *N*-клавішного натискання
1	0	Сканування матриці датчиків
1	1	Режим стробованого введення

Розряд *S* визначає режим сканування: за $S = 0$ — сканування у режимі чотирирозрядного двійкового лічильника; за $S = 1$ — сканування у режимі інверсного дешифратора по лініях *S*3—*S*0. Якщо сканування здійснюється у режимі дешифратора, то дисплей містить не більше ніж 4 символи, а клавіатура — не більш як $8 \times 4 = 32$ клавіш.

У керуючому слові *ініціалізації опорної частоти* (рис. 5.21) розряди *D*4—*D*0 визначають коефіцієнт *PPPPP* ділення частоти зовнішнього синхросигналу *CLK* для отримання внутрішнього опорного сигналу з частотою не більше ніж 100 кГц. Після скидання ВІС сигналом *RESET* встановлюється максимальний коефіцієнт *PPPPP* =11111.

Приклад 5.9. Запрограмувати ВІС контролера клавіатури й індикації для роботи з клавіатурою 8 × 8 клавіш у режимі *N*-клавішного натискання та з дисплеєм з 8 символів у режимі введення з лівого

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	P	P	P	P	P

Рис. 5.21. Формат керуючого слова ініціалізації опорної частоти

боку. Згідно зі схемою підключення (див. рис. 5.19) ВІС КР580ВВ79 має адреси 00*H* для даних і 02*H* для запису керуючих слів та читання статусної інформації. Частота імпульсів на вході *CLK* становить 2 МГц.

Визначимо керуючі слова. За умовою прикладу та рис. 5.20 керуюче слово ініціалізації клавіатури і дисплея має вигляд

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	0

і дорівнює 02*H*. Керуюче слово ініціалізації опорної частоти (див. рис. 5.20) таке:

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	1	0	1	0	0

і дорівнює 34*H*. Значення *PPPPP* = 10100_2 визначає двійковий код коефіцієнта ділення імпульсів з частотою 2 МГц для забезпечення внутрішньої частоти 100 кГц (2000/100 = 20 = 10100_2).

Програма ініціалізації ВІС:

```
MOV  AL, 02    ; Формування першого керуючого слова режиму в AL
OUT  02,AL     ; Виведення в інтерфейс
MOV  AL, 34H   ; Формування другого керуючого слова режиму в AL
OUT  02,AL     ; Виведення в інтерфейс.
```

Після такої послідовності команд інтерфейс клавіатури та індикації готовий до роботи у запрограмованому режимі.

З'єднання ВІС інтерфейсу з клавіатурою та індикацією зображено на рис. 5.22.

Штриховою лінією показано під'єднання зовнішнього дешифратора *DC* до ВІС. Якщо клавіатура містить менше ніж 4 × 8 клавіш, а кількість символів дисплея менше ніж 4, то сигнали на виводах *S*3—*S*0 можна безпосередньо використовувати для керування клавіатурою й індикацією, оскільки на виводах *S*3—*S*0 формуються сигнали дешифратора з чотирма виходами. Під'єднання зовнішнього дешифратора (до 16 виходів) дає змогу керувати клавіатурою 16 × 8 клавіш та 16 символами дисплея.

Функціонування блока інтерфейсу індикації. Після запису керуючого слова ініціалізації клавіатури та дисплея (рис. 5.20) блок інтерфейсу індикації встановлюється в один з чотирьох режимів, які визначаються розрядами *D*3 і *D*4. В усіх режимах для висвітлення символу на індикації треба завантажити

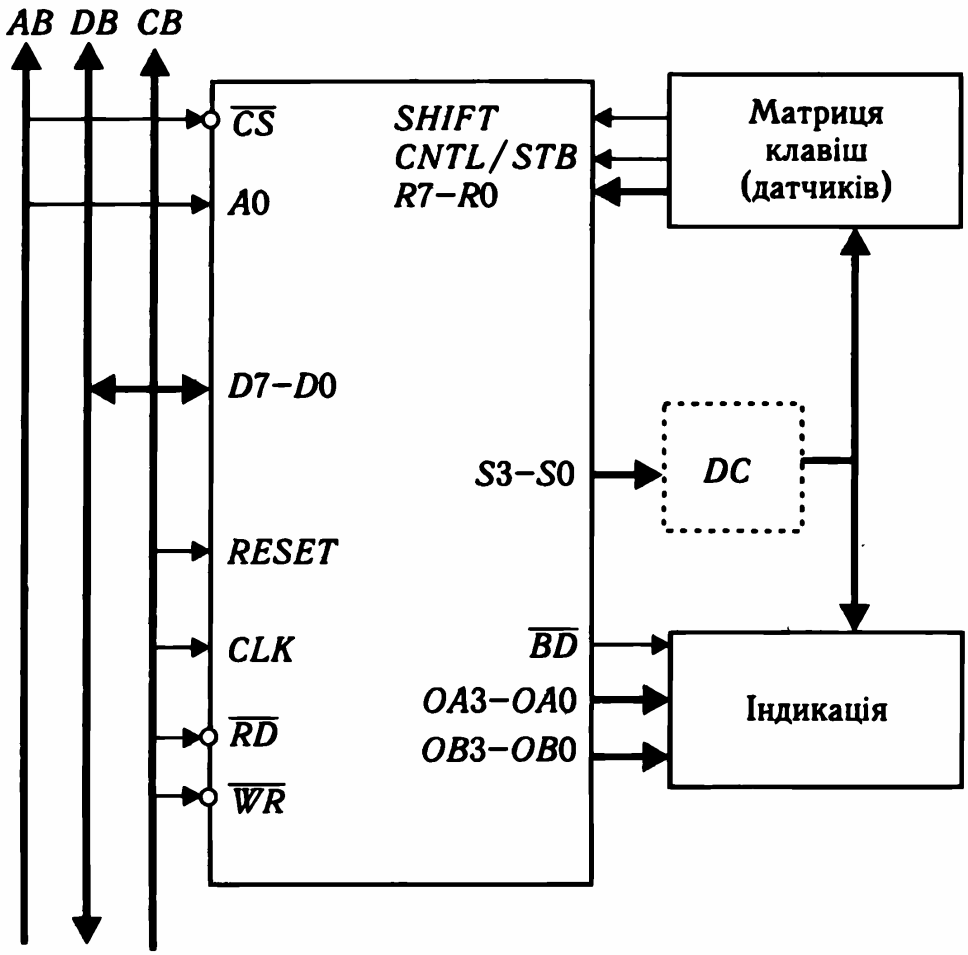


Рис. 5.22. Структурна схема з'єднання ВІС з клавіатурою та індикацією

Рис. 5.23. Формат керуючого слова запису в ОЗП індикації

D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	I	A	A	A	A

керуюче слово *запису в ОЗП індикації* (рис. 5.23) з адресою із $A0 = 1$, а потім завантажити дані з адресою з $A0 = 1$. Під час індикації дані з ОЗП передаються на 8 ліній $OA3-OA0$, $OB3-OB0$. Дані можуть бути подані семисегментним кодом за безпосереднього з'єднання індикаторів з лініями $OA3-OA0$, $OB3-OB0$ або двома чотирирозрядними кодами під час підключення зовнішніх шифраторів.

Розряди $D3-D0$ керуючого слова (рис. 5.23) містять адресу *AAAA* позиції дисплея, яка має бути прочитана. Розряд $D4$ містить ознаку автоінкрементної адресації I. Якщо $I = 1$, адреса буде інкрементуватися після кожної операції читання.

Для зчитування даних з ОЗП індикації слід завантажити керуюче слово ***зчитування з ОЗП індикації*** (рис. 5.24) за $A0 = 1$, а потім зчитати інформацію з ОЗП за $A0 = 0$.

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	*I*	*A*	*A*	*A*	*A*

Рис. 5.24. Формат керуючого слова зчитування з ОЗП індикації

Призначення розрядів керуючого слова (див. рис. 5.24) аналогічне призначенню розрядів керуючого слова (див. рис. 5.23). Поле *AAAA* визначає адресу позиції в ОЗП індикації, яка має бути зчитана.

Якщо дисплей містить 8 символів, то блок керування мікросхеми КР580ВВ79 сканує дисплей за 5,1 мс за внутрішньої частоти 100 кГц, якщо 16 символів, — то за 10,23 мс. Процес сканування дисплея полягає у видаванні у вихідний регістр індикації *OUT RG* (див. рис. 5.18) вмісту кожної комірки ОЗП індикації. Функціонування блока інтерфейсу індикації залежить від способу видавання кодів сканування *S*3—*S*0 (див. рис. 5.18). У процесі сканування в *режимі інверсного дешифратора* інформація з'являється лише у перших чотирьох знакомісцях дисплея, у режимі чотирирозрядного двійкового лічильника під час використання зовнішнього дешифратора — на 16. Одночасно зі зміною станів лічильника сканування і вмісту вихідного регістра індикації на виводі $\overline{BD}$ з'являється сигнал логічного нуля тривалістю 150 мкс, що використовується для гасіння індикації зі зміною символів.

У режимах виведення інформації на індикацію із введенням нових символів *з лівого боку* кожному знакомісцю дисплея відповідає один байт в ОЗП індикації. Комірці ОЗП з нульовою адресою відповідає крайнє ліве знакомісце, а комірці з адресою 7 (або 15) — крайнє праве в індикації відповідно на 8 або 16 символів. У режимах виведення інформації з введенням нових символів *з правого боку* код нового символу записується у комірку ОЗП з нульовою адресою, при цьому раніше записана інформація зсувається вліво. Слід зазначити, що у цьому режимі немає прямої відповідності між адресою комірки ОЗП і знакомісцем індикації.

Програмним способом можна заборонити видавання однієї чи обох тетрад вмісту вихідних регістрів. Вигляд керуючого слова *заборони запису ОЗП індикації-гасіння* показано на рис. 5.25.

Байти ОЗП індикації поділяють на тетради *A* (старша) і *B* (молодша). Розряди *D*3 (*IWA*) і *D*2 (*IWB*) — це біти забо-

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	*X*	*IWA*	*IWB*	*BLA*	*BLB*

Рис. 5.25. Формат керуючого слова заборони запису у ОЗП індикації-гасіння

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	CD	BC_1	BC_2	CF	CA

Рис. 5.26. Формат керуючого слова встановлення коду бланкування, сканування ОЗП індикації та скидання байта стану

роли запису інформації у тетради *A* і *B* ОЗП індикації, розряди *D*1 (*BLA*) і *D*3 (*BLB*) — біти гасіння або бланкування (встановлення спеціального коду, наприклад, коду пробілу). Керуюче слово (див. рис. 5.25) дозволяє маскувати одну з тетрад у випадку подвійного чотирипозиційного дисплея. У разі заборони запису в одну з тетрад тривалість низького рівня сигналу гасіння становить не менше ніж 150 мкс, а в разі заборони запису в обидві тетради визначається тривалістю дії керуючого слова.

Для *встановлення коду бланкування* і *сканування ОЗП індикації та скидання байта стану* використовується керуюче слово, формат якого показано на рис. 5.26.

Розряди *D*3 (BC_1) та *D*2 (BC_2) дозволяють обрати один з кодів бланкування:

Розряди		Коди бланкування
BC_1	BC_2	
0	X	00
1	0	20*H* (код пробілу)
1	1	0*FFH*

Із встановленням розряду *D*4 (*CD*) відбувається процедура скидання ОЗП індикації відповідним заповненням кодами бланкування. Після встановлення розряду *D*1(*CF*) скидання байта стану та сигналу переривання і покажчик пам'яті матриці клавіатури встановлюється на нульовий рядок. Дія розряду *D*0(*CA*) аналогічна одночасній дії розрядів *D*4 і *D*1.

Приклад 5.10. Інтерфейс клавіатури та індикації запрограмовано на режим сканування 8-символьного дисплея із введенням з лівого боку. До ВІС інтерфейсу під'єднано три семисегментних індикатори в позиціях 0, 1, 2. Навести функціональну схему під'єднання дисплея і написати програму видавання на дисплей вмісту трьох 8-бітових комірок пам'яті з початковою адресою *DS* : *SI*, в яких записано семисегментні коди. Адреси ВІС контролера клавіатури й індикації такі, як і в прикладі 5.9.

Функціональну схему під'єднання дисплея зображено на рис 5.27.

Дисплей складається з трьох семисегментних індикаторів на зразок АЛС321Б, схема кожного з яких містить вісім світлодіодів, з'єднаних

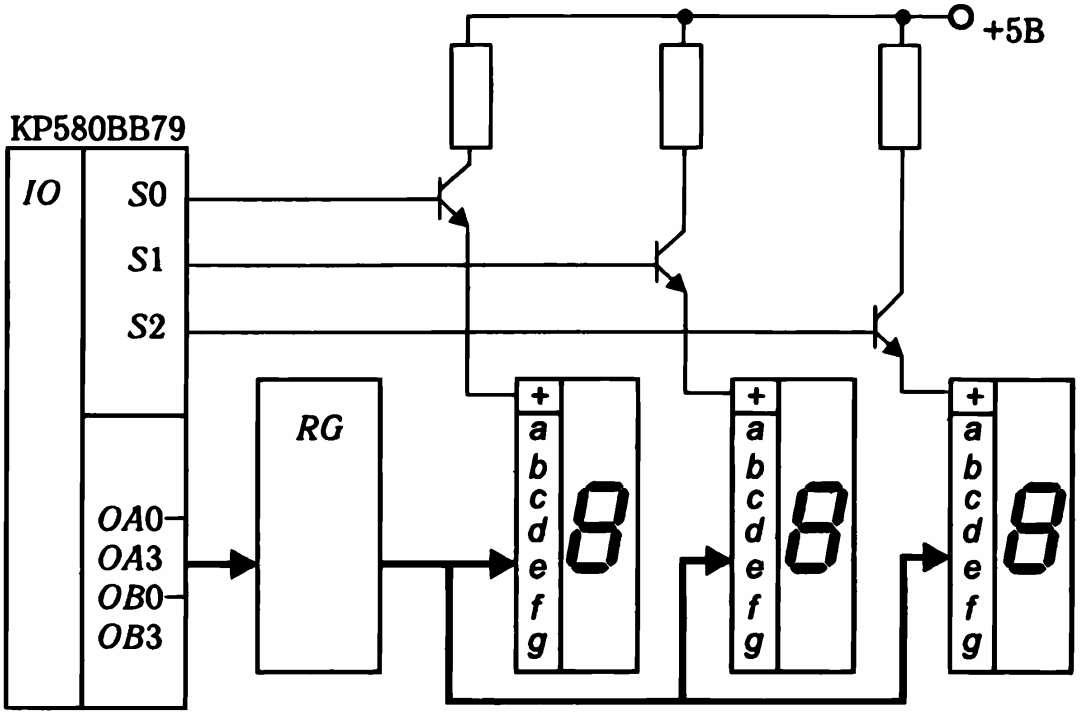


Рис. 5.27. Функціональна схема під'єднання дисплея

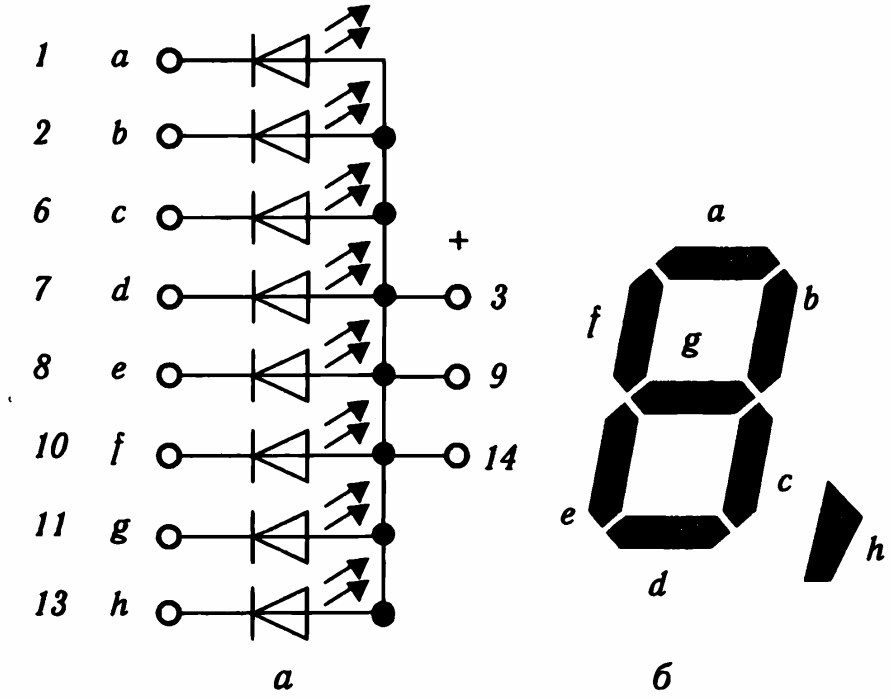


Рис. 5.28. Семисегментний індикатор АЛС321Б:
а — принципова схема; *б* — розміщення світлодіодів

за схемою зі спільним анодом (рис. 5.28, *а*). Світлодіод вмикається після надходження на входи *a*—*g* сигналу низького рівня (рис. 5.28, *б*), а на спільний анод — сигналу високого рівня.

Використання семисегментних індикаторів цього типу потребує подання інверсних семисегментних кодів. Індикатори гасяться з надходженням коду бланкування 0*FFH*.

Програма виведення символів на індикацію починається з гасіння дисплея записуванням коду $0FFH$ в ОЗП індикації. Керуюче слово встановлення коду бланкування, сканування ОЗП індикації та скидання байта стану (див. рис. 5.26) має містити значення $CD = 1$, $BC1 = BC2 = 1$. Тоді значення керуючого слова дорівнює $0DCH$.

Інформація на дисплей виводиться після завантаження керуючого слова запису в ОЗП індикації (див. рис. 5.25), в якому $AAAA = 0000$, $I = 1$. Тоді значення керуючого слова дорівнює $70H$.

Програма виведення символів на індикацію така:

```
      MOV   AL, 0DCH   ; Формування керуючого слова встановлен-
                       ; ня коду
                       ; Бланкування, сканування ОЗП індикації
                       ; і скидання
                       ; байта стану
      OUT   02, AL     ; Виведення на інтерфейс
      CALL  DELAY1     ; Затримка на час бланкування
      MOV   AL, 70H    ; Формування керуючого слова запису
                       ; в ОЗП індикації
      OUT   02, AL     ; Виведення на інтерфейс
      MOV   CX, 3      ; Завантаження лічильника байтів
M0:   MOV   AL, [SI]   ; Зчитування з ОЗП індикації
      OUT   00, AL     ; Виведення даних на інтерфейс
      INC   SI         ; Збільшення адреси на одиницю
      LOOP  M0
      CALL  DELAY2     ; Затримка на час сканування
```

Оскільки нові значення на індикацію можна передавати лише після затримки на час сканування даних в ОЗП дисплея (5,1 мс), то наведена програма містить підпрограму часової затримки $DELAY2$.

Функціонування блока інтерфейсу клавіатури (датчиків). Усі режими роботи блока інтерфейсу клавіатури (датчиків), що визначаються розрядами $D1$ та $D2$ керуючого слова ініціалізації клавіатури і дисплея (див. рис. 5.20), можна поділити на три групи:

- опитування матриці клавіатури;
- опитування матриці датчиків;
- введення даних за стробом.

У ***режимі опитування матриці клавіатури*** (за $D1 = D2 = 0$ та за $D1 = 1$, $D2 = 0$) натискання будь-якої клавіші ініціює генерацію високого рівня сигналу переривання на виводі IRQ, а код натиснутої клавіші записується в ОЗП клавіатури (датчиків). Під час опитування матриці клавіатури функціонує схема усунення брязкоту контактів. Звернення до ОЗП відбувається за принципом черги: код, записаний в ОЗП першим, зчитується з нього першим. Щоб зчитати код клавіші з ОЗП, треба завантажити в інтерфейс клавіатури та індикації керуюче слово *читання ОЗП клавіатури (датчиків)* (рис. 5.29) за $A0 = 1$, а потім зчитати дані з ОЗП за $A0 = 0$.

Розряд $D4$ (див. рис. 5.29) містить ознаку автоінкрементної адресації I, розряди $D2-D0$ — адресу AAA байта ОЗП клавіатури (датчиків), що має бути зчитаним. Якщо біт I встановлено, то наступні команди читання даних зумовлюватимуть автоматичне інкрементування адреси. Для читання вмісту всього ОЗП необхідно завантажити керуюче слово читання ОЗП клавіатури (датчиків) за $I = 1$, а після цього 8 разів зчитати дані.

Формат даних під час читання ОЗП клавіатури (датчиків) наведено на рис. 5.30. У розрядах $D5-D3$ розміщується номер рядка матриці натиснутої клавіші — значення розрядів $S2-S0$ лічильника сканування; у розрядах $D2-D0$ розміщений номер стовпця матриці натиснутої клавіші — значення розрядів $R2-R0$. Розряди $D7-D6$ можна використовувати у разі введення з розширеної клавіатури — в них записується стан додаткових клавіш, з'єднаних з виводами *CNTL* і *SHIFT*.

У процесі зчитування даних з ОЗП клавіатури (датчиків) відбувається скидання сигналу переривання *IRQ*, але якщо ОЗП клавіатури (датчиків) містить ще не зчитані дані, на виводі *IRQ* знову генерується сигнал високого рівня.

Режим опитування матриці клавіатури має:

- режим одиничного натискання клавіш із забороною введення кодів після натискання двох або більше клавіш ($D1 = D2 = 0$);
- N-клавішного натискання із дозволом введення кодів у разі натискання N клавіш ($D1 = 1$, $D2 = 0$).

У *режимі одиничного натискання*, якщо натиснуто дві або більше клавіш, в ОЗП клавіатури (датчиків) записується код лише однієї з клавіш — першої натиснутої або тієї, що опитується першою.

У *режимі N-клавішного натискання* в ОЗП заносяться коди всіх натиснутих клавіш по черзі їх опитування під час

D7	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*
0	1	0	*I*	*x*	*A*	*A*	*A*

Рис. 5.29. Формат керуючого слова читання ОЗП клавіатури (датчиків)

D7	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*
CNTL	*SHIFT*	*S2*	*S1*	*S0*	*R2*	*R1*	*R0*

Рис. 5.30. Формат даних під час читання ОЗП клавіатури (датчиків)

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	E	x	x	x	x

Рис. 5.31. Формат керуючого слова скидання переривання-встановлення режиму помилки сканування

D7	D6	D5	D4	D3	D2	D1	D0
DU	S/E	O	U	F	N2	N1	N0

Рис. 5.32. Формат байта стану

сканування матриці клавіатури. У цьому режимі можна запрограмувати ВІС інтерфейсу на спеціальний *режим помилки сканування клавіатури* встановленням одиничного значення розряду $D4(E)$ у керуючому слові скидання перериванння-встановлення режиму помилки сканування (рис. 5.31). Це керуюче слово також встановлює L-рівень сигналу переривання на лінії IRQ.

У разі одночасного натискання кількох клавіш встановлюється прапорець помилки у байті стану і генерований високий рівень сигналу на виводі IRQ. *Формат байта стану* наведено на рис. 5.32. Одиничне значення розряду $D7(DU)$ вказує на недоступний дисплей, тобто на те, що не закінчена операція очищення ОЗП індикації, одиничне значення розряду $D6(S/E)$ — на те, що датчик замкнений (у режимі опитування матриці датчиків) або на помилку багатоклавішного натискання (у режимі опитування клавіатури), одиничне значення розряду $D5(O)$ — на помилку переповнення. Цей розряд встановлюється тоді, коли робиться спроба запису в заповнену пам'ять клавіатури. Одиничне значення розряду $D4(U)$ вказує на помилку спустошення і встановлюється тоді, коли робиться спроба зчитування даних з порожньої ОЗП клавіатури (датчиків). Одиничне значення розряду $D3(F)$ вказує на заповненість ОЗП клавіатури (датчиків). Розряди $D2-D0$ $(N2-N0)$ визначають кількість символів в ОЗП клавіатури (датчиків).

У ***режимі опитування матриці датчиків*** зміна стану одного з датчиків ініціює генерацію високого рівня сигналу на виводі IRQ. При цьому значення розрядів $R7-R0$ безпосередньо записуються в ОЗП клавіатури (датчиків) без передавання керування схемі усунення брязкоту контактів.

У ***режимі введення за стробом*** значення розрядів $R7-R0$ записуються в ОЗП клавіатури (датчиків), але введення стробується сигналом на виводі $CNTL/STB$.

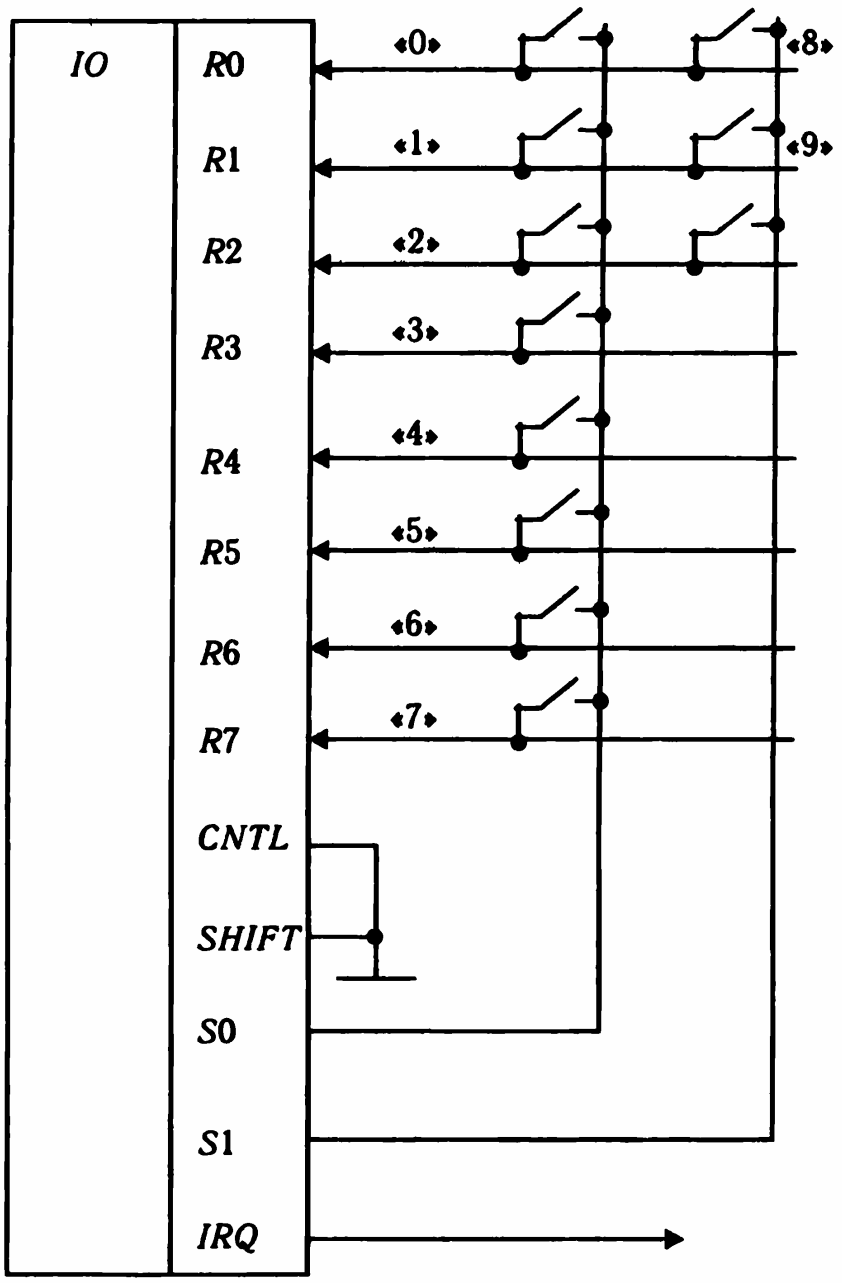


Рис. 5.33. Функціональна схема з'єднання ВІС з клавіатурою

Приклад 5.11. Навести функціональну схему з'єднання ВІС з інтерфейсом клавіатури, яка містить клавіші десяти цифр та клавішу *ENTER*. Визначити коди клавіш. Інтерфейс клавіатури та індикації запрограмовано на режим опитування матриці клавіатури із забороною введення кодів після натискання N клавіш.

Функціональну схему з'єднання ВІС з клавіатурою зображено на рис. 5.33.

За такого з'єднання коди клавіш, які записуються в ОЗП клавіатури, визначають за допомогою рис. 5.30. Для цифр від 0 до 9 коди збігаються з цифрами, позначеними в лапках (див. рис. 5.33). Код клавіші *ENTER* визначається як 00 001 010 і дорівнює $0AH$.

5.4. Програмований таймер

Програмований таймер (ПТ) КР1810ВИ54 призначений для організації роботи МП систем та формування сигналів з різними часовими і частотними характеристиками. Структурну схему ВІС зображено на рис. 5.34, а умовне позначення — на рис. 5.35.

Схема таймера містить блок керування читанням-записом *RWCU* з регістром керуючого слова *RCW*, тристабільний буфер даних *BD*, три канали на базі 16-розрядних від'ємних лічильників *CT*0—*CT*2, причому кожний канал містить лічильник, вхідні та вихідні буферні регістри. Лічильники можуть

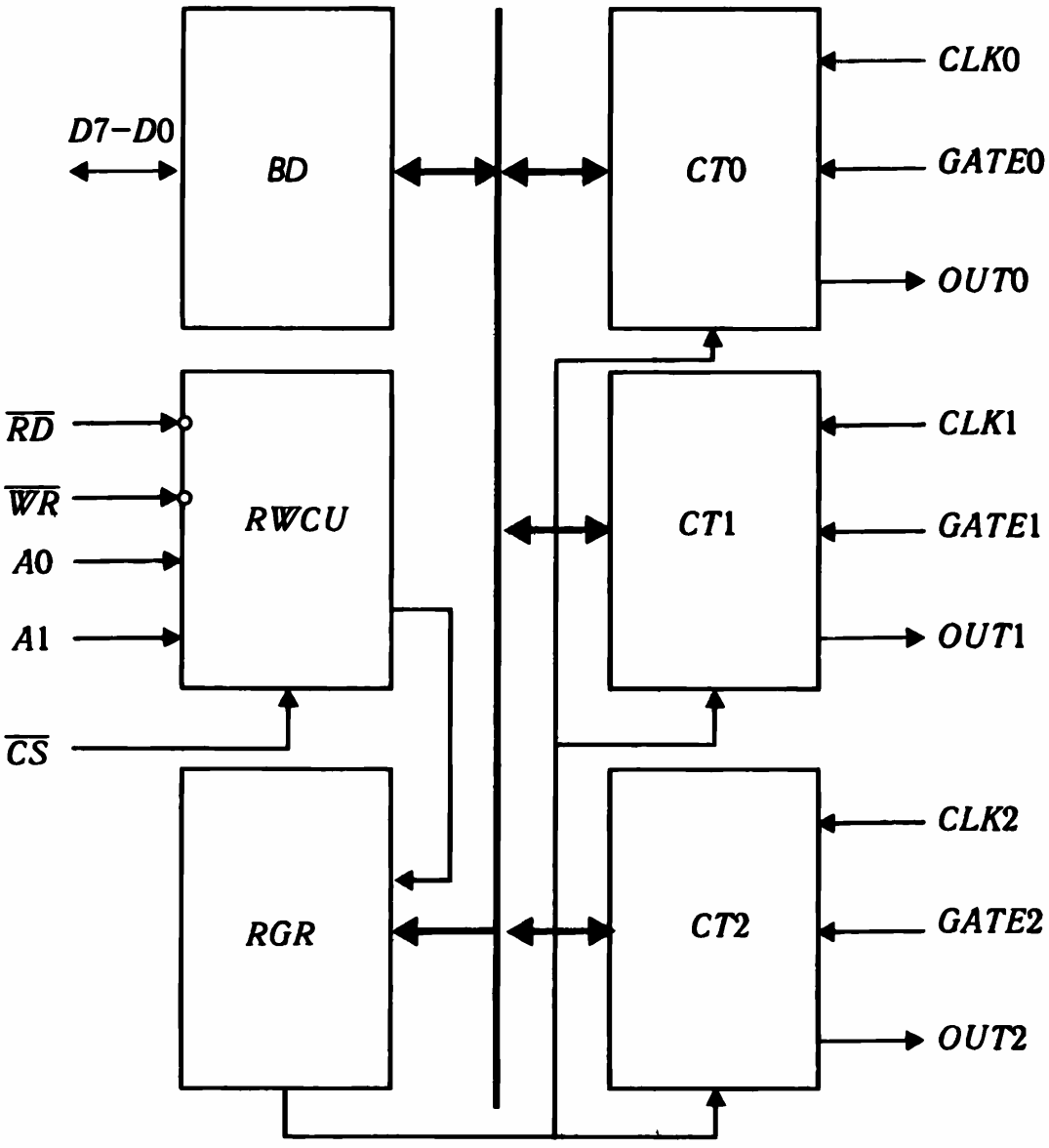


Рис. 5.34. Структурна схема ВІС таймера КР580ВИ54:
CLK — входи тактових (лічильних) імпульсів; *GATE* — входи дозволу лічення (залежить від режиму роботи каналу); *OUT* — виходи лічильника

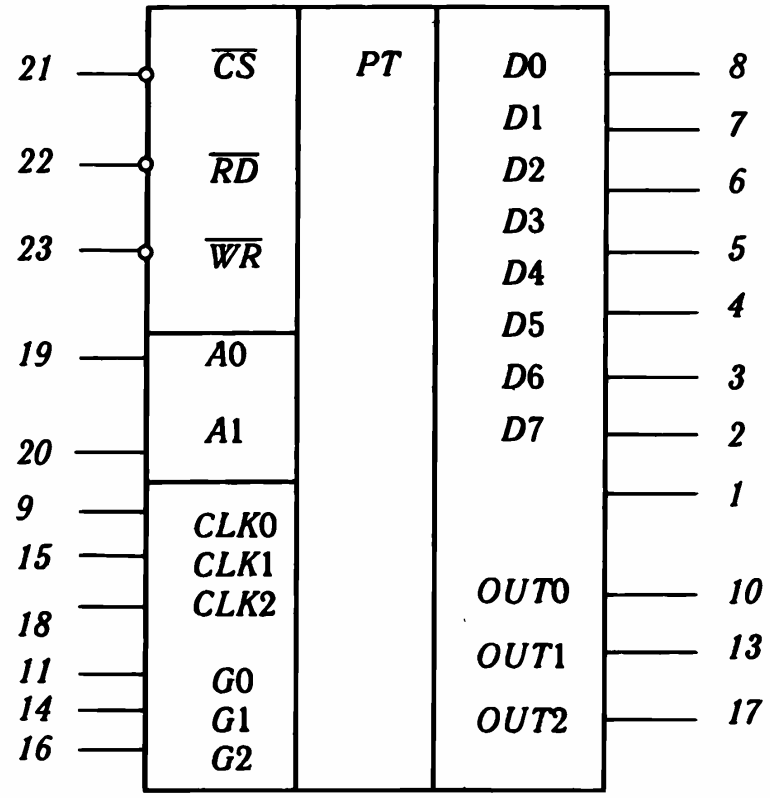


Рис. 5.35. Умовне позначення ВІС таймера КР580ВИ54

працювати у двійковому або двійково-десятковому коді. Максимальна частота лічильника становить 2 МГц для таймера КР580ВИ53 та 5МГц для таймера КР1810ВИ54

У табл. 5.6 наведено призначення виводів ПТ КР580ВИ54.

Розряди $A1$, $A0$ (див. рис. 5.34) обирають звернення до лічильників або до регістра керуючого слова RCW:

$A1$	$A0$	Звернення
0	0	$CT0$
0	1	$CT1$
1	0	$CT2$
1	1	RCW

Сигнали керування роботою ВІС WR, RD, CS подаються на блок $RWCU$ і разом з адресними розрядами $A0$, $A1$ задають вид виконуваної операції згідно з табл. 5.7.

Узагальнену схему під'єднання програмованого таймера до шин мікропроцесора зображено на рис. 5.36. Наприклад, на адресні лінії $A1$, $A0$ можна під'єднати лінії $A2$, $A1$ шини адрес, а на вхід $\overline{CS}$ подати сигнал з виходу дешифратора (див. рис. 5.9 або рис. 5.19).

Таблиця 5.6. **Призначення виводів ВІС таймера КР580ВИ54**

Позначення	Номер виводу	Призначення виводів
$D7-D0$	1–8	Мультиплексована шина даних (DB), по якій з розподілом у часі передаються дані
$\overline{CS}$	21	Вибірка кристалу; за $\overline{CS}=0$ робота ВІС дозволяється
$\overline{RD}$	22	Читання. Сигнал $\overline{RD}=0$ налагоджує вхідний буфер на виведення, за якого програмований таймер видає інформацію у МП
$\overline{WR}$	23	Запис. Сигнал $\overline{WR}=0$ налагоджує вхідний буфер на введення, за якого програмований таймер приймає інформацію від МП
$A0, A1$	19; 20	Адресні входи, за якими відбувається адресація до одного з трьох каналів таймера
$CLK2-CLK0$	9; 15; 18	Вхід тактових сигналів для керування лічильником-таймером. Зріз сигналу на вході CLK призводить до зменшення вмісту лічильника таймера на одиницю
$GATE2-GATE0$	11; 14; 16	Входи дозволу лічби
$OUT2-OUT0$	10; 13; 17	Виходи лічильника-таймера

Таблиця 5.7. **Вид операції програмованого таймера залежно від сигналів керування та адресних розрядів**

Операція	Сигнали керування				
	$\overline{WR}$	$\overline{RD}$	$\overline{CS}$	$A0$	$A1$
Запис керуючого слова в RCW	0	1	0	1	1
Завантаження					
$CT0$	0	1	0	0	0
$CT1$	0	1	0	0	1
$CT2$	0	1	0	1	0
Читання					
$CT0$	1	0	0	0	0
$CT1$	1	0	0	0	1
$CT2$	1	0	0	1	0
Від'єднання програмованого таймера від шини	1	1	0	x	x

Примітка. x — будь-яке значення (0 або 1).

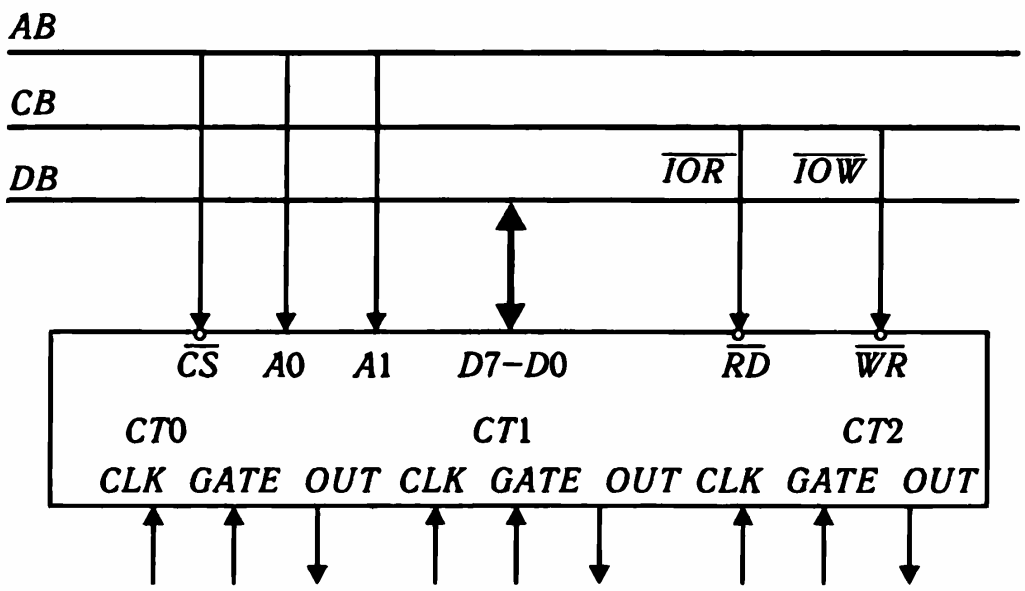


Рис. 5.36. Під'єднання програмованого таймера до шин мікропроцесора

D7	D6	D5	D4	D3	D2	D1	D0
CNT1	CNT0	RW1	RW0	M2	M1	M0	K

Рис. 5.37. Формат керуючого слова програмованого таймера

Установлення режиму роботи кожного каналу програмованого таймера здійснюється програмно — записуванням керуючого слова (рис. 5.37) і початкового вмісту лічильника.

За значеннями розрядів $D7$ ($CNT1$) та $D6$ ($CNT0$) вибирають лічильник (табл. 5.8).

За значеннями розрядів $D5$ ($RW1$) та $D4$ ($RW2$) вибирають спосіб читання-запису (табл. 5.9).

За значеннями розрядів $D3-D1$ ($M2-M0$) вибирають один з шести режимів роботи лічильника (табл. 5.10).

Таблиця 5.8. **Вибирания лічильника**

Розряд		Лічильник
D7	D6	
0	0	CT0
0	1	CT1
1	0	CT2
1	1	Заборонена комбінація для таймера КР580ВИ53 та команда читання слову стану для таймера КР1810ВИ54

Таблиця 5.9. **Спосіб читаиия-запису**

Розряд		Спосіб читання-запису
*D*5	*D*4	
0	0	Читання вмісту лічильника
0	1	Запис лише молодшого байта
1	0	Запис лише старшого байта
1	1	Запис молодшого, а потім старшого байтів

Таблиця 5.10. **Режими роботи програмованого таймера**

*M*2	*M*1	*M*0	Режим	*M*2	*M*1	*M*0	Режим
0	0	0	0	*x*	1	1	3
0	0	1	1	1	0	0	4
x	1	0	2	1	0	1	5

Розряд *D*0 (*K*) визначає спосіб кодування:
- *D*0 = 0 — двійковий лічильник;
- *D*0 = 1 — двійково-десятковий лічильник.

Приклад 5.12. Запрограмувати лічильник 0 в режим 1. Адреса лічильника 0 — 10*H*, регістра керуючого слова — 16*H*.

Визначимо керуюче слово:

0011 0010 = 32*H*

Програма матиме такий вигляд:

```
MOV AL, 32H              ; Формування керуючого слова
OUT 16H, AL              ; Виведення в RCW
MOV AL, «молодший байт»  ; Завантаження молодшого байта коду
OUT 10H, AL              ; Попереднє встановлення
MOV AL, «старший байт»   ; Завантаження старшого байта коду
OUT 10H, AL              ; Попереднє встановлення
```

Порядок програмування каналів таймера надзвичайно гнучкий. Можна записати керуючі слова режимів у всі канали, а потім у довільному порядку завантажувати коди попереднього встановлення, а можна запрограмувати окремо кожний канал (як у прикладі 5.12).

У процесі роботи програмованого таймера вміст будь-якого з лічильників можна прочитати двома способами:

призупинити роботу лічильника надходженням сигналу *GATE* = 0 або блокуванням тактових імпульсів, а потім прочитати вміст лічильника, починаючи з молодшого байта, за допомогою двох команд введення. Перша команда введення прочитає молодший байт, друга — старший;

записати у програмований таймер керуюче слово, що містить нулі в розрядах *D*4, *D*5 (нулі в цих розрядах указують на виконання операції «замкнення» вмісту лічильника у вихідному регістрі каналу в момент запису керуючого слова). Потім прочитати вміст лічильника за допомогою команд введення.

Приклад 5.13. Прочитати вміст лічильника *CT*0 і записати його у регістр *BX*.

Візьмемо адреси таймера такі, як у прикладі 5.12. Запрограмувати лічильник 0 у режимі 1.

Визначимо керуюче слово для фіксації вмісту лічильника:

0000 00102 = 02*H*

Програма буде така:

```
MOV AL, 02H      ; Формування керуючого слова
OUT 16H, AL      ; Виведення в RCW
IN AL, 10H       ; Читання молодшого байта
MOV BL, AL       ; Пересилання молодшого байта у BL
IN AL, 10H       ; Читання старшого байта
MOV BH, AL       ; Пересилання старшого байта у BH
```

Отже, після виконання програми у *BX* буде вміст лічильника на момент його читання, а лічильник продовжуватиме лічення.

Крім того, у ВІС К1810ВИ54 можна прочитати слова стану лічильника. Для цього треба записати керуюче слово (рис. 5.38):

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
1	1	*COUNT*	*STAT*	*CT*2	*CT*1	*CT*0	0

Рис. 5.38. Керуюче слово ВІС К1810ВИ54:
*CT*0, *CT*1, *CT*2 — вибір лічильника

Лічильник вибирається для запису одиниці у відповідний двійковий розряд. Значення *STAT* = 0 вказує на те, що буде прочитано слово стану каналу, зазначеного в розрядах *D*3—*D*1. Значення *COUNT* = 0 свідчить про те, що буде запам'ятовано вміст лічильників, зазначених у розрядах *D*3—*D*1 та вихідних регістрах каналів.

Слово стану каналу має вигляд, показаний на рис. 5.39.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
OUT	*FN*	*RW*1	*RW*0	*M*2	*M*1	*M*0	*K*

Рис. 5.39. Вигляд слова стану каналу:
OUT — стан виходу *OUT* (0,1); *FN* — прапорець перевантаження (*FN* = 1, якщо було перезавантаження коду переднього встановлення); *RW*1, *RW*0, *M*2, *M*1, *M*0, *K* — розряди (дублюють розряди керуючого слова, див. рис. 5.38)

Під час запису керуючого слова в лічильник завантажується спочатку молодший, а потім старший байт коду переднього встановлення. Надалі робота таймера залежить від обраного режиму роботи.

Режими роботи таймера. Лічильники таймера можуть працювати у таких шести режимах: 0 — програмована затримка; 1 — програмований мультивібратор; 2 — програмований генератор тактових імпульсів; 3 — генератор прямокутних сигналів; 4 — програмно-керований строб; 5 — апаратно-керова-

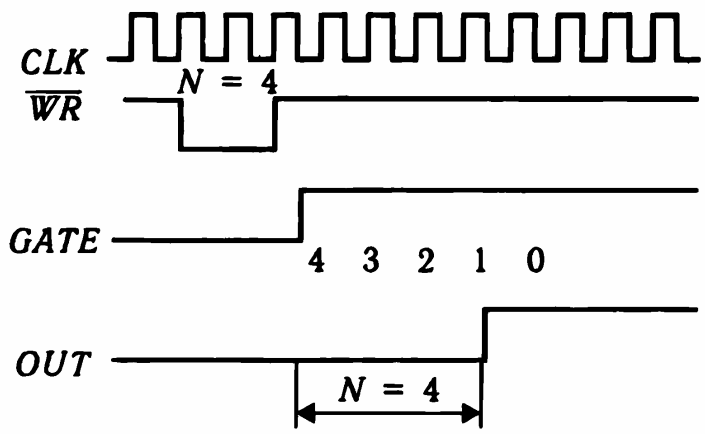


Рис. 5.40. Режим програмованої затримки

ний строб. Вплив сигналу *GATE* на відповідний лічильник залежить від режиму роботи.

Режим 0. Програмована затримка. У цьому режимі (рис. 5.40) на виході вибраного каналу таймера формується сигнал *H*-рівня з програмно-керованою затримкою. Затримка відлічується від заднього фронту першого імпульсу *CLK* після запису молодшого байта коду перестановки константи. Після запису керуючого слова на виході *OUT* вибраного каналу таймера встановлюється сигнал *L*-рівня. Такий самий стан зберігається під час запису молодшого байта константи. Якщо під час лічення *GATE* = 0, то лічення припиняється, а з появою *GATE* = 1 — відновлюється з перерваного значення. Після закінчення лічення на виході *OUT* встановлюється сигнал *H*-рівня. Завантаження у лічильник нового значення молодшого байта у процесі лічення його припиняє, а після завантаження старшого байта починається новий цикл лічення.

Режим 1. Програмований мультивібратор. На виході лічильника формується імпульс *L*-рівня з програмно-керованою тривалістю, причому точкою початку відліку є задній фронт першого імпульсу *CLK* після появи сигналу *GATE* (рис. 5.41).

Якщо значення сигналу *GATE* = 1, на виході *OUT* формується імпульс *L*-рівня тривалістю *N* періодів тактових імпульсів *CLK*. Завантаження у процесі лічення нового значення *N* поточного режиму лічення не змінює.

Мультивібратор автоматично перезапускається після кожного переднього фронту сигналу *GATE*.

Режим 2. Програмований генератор тактових імпульсів. У цьому режимі (рис. 5.42) обраний канал здійснює розподіл частоти імпульсів *CLK* на програмно-керований коефіцієнт *N*, тобто програмований таймер генерує

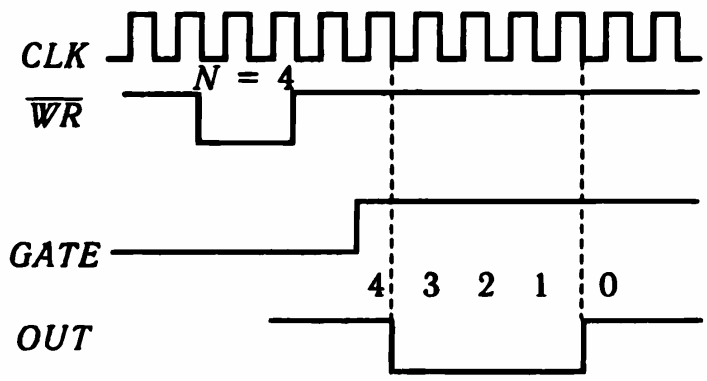


Рис. 5.41. Режим програмованого мультивібратора

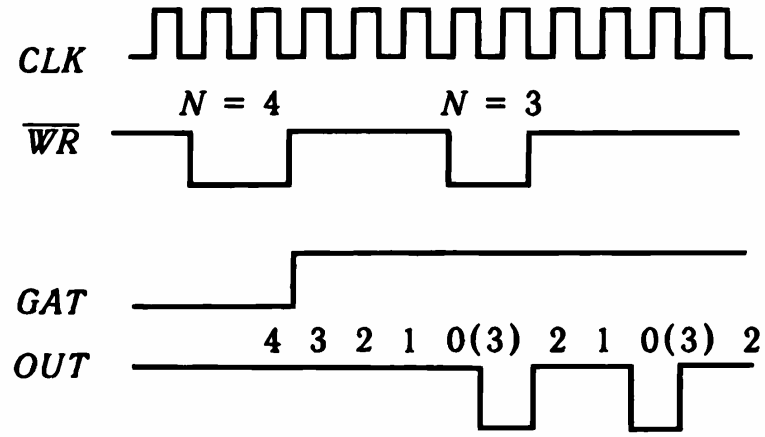


Рис. 5.42. Режим програмованого генератора тактових імпульсів

періодичний сигнал з частотою у N разів меншою, ніж частота тактових імпульсів CLK.

Вихідний сигнал L-рівня встановлюється на останньому такті періоду. Завантаження лічильника новим значенням N у процесі лічби призводить до зміни розміру періоду. Сигнал $GATE$ можна використовувати для зовнішньої синхронізації програмованого таймера, оскільки значення $GATE = 0$ забороняє лічення, встановлюючи значення сигналу $OUT = 1$, а значення $GATE = 1$ починає лічення спочатку.

Режим 3. Генератор прямокутних імпульсів. Обраний канал формує прямокутні імпульси з програмно-керованим періодом. Дія сигналу $GATE$ аналогічна режиму 0. За парного значення N на виході лічильника генерується сигнал H-рівня впродовж першої половини періоду і сигнал L-рівня впродовж другої половини. У разі непарного N тривалість сигналу H-рівня на один такт більша, ніж для сигналу L-рівня. У режимі 3 число $N = 3$ завантажувати у лічильник не дозволяється. Часові діаграми для цього режиму зображено на рис. 5.43.

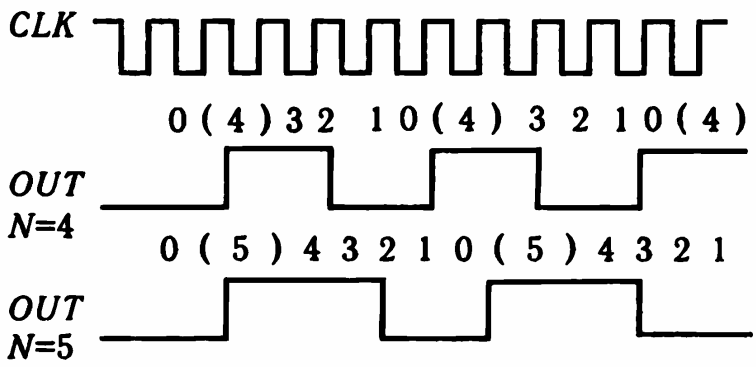


Рис. 5.43. Режим генератора прямокутних імпульсів

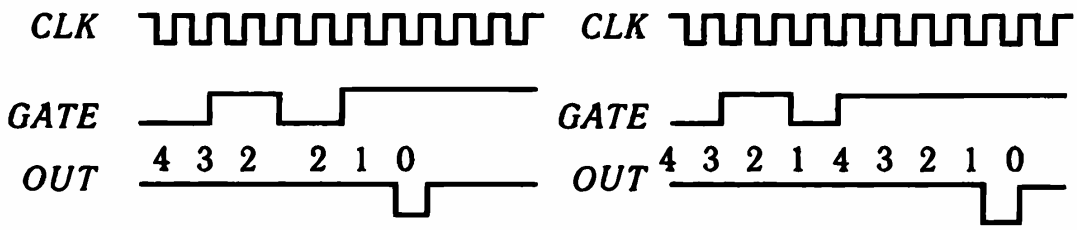


Рис. 5.44. Діаграми роботи таймера:
а — для таймера КР580ВИ53; *б* — для таймера КР1810ВИ54

Режим 4. Програмовано-керований строб. У цьому режимі на виході лічильника формується строб *L*-рівня тривалістю T_{CLK} з програмно-керованою затримкою щодо моменту запису молодшого байта команди. Перезавантаження молодшого байта у процесі лічби не впливає на поточне лічення, а завантаження старшого байта починає новий цикл лічення.

Режим 5. Апаратно-керований строб. Цей режим аналогічний режиму 4. Його відмінність від режиму 4 полягає в тому, що початком відліку програмно-керованої затримки є передній фронт сигналу *GATE*. Запуск лічильника здійснюється переднім фронтом сигналу *GATE*. Завантаження у лічильник нового значення *N* у процесі лічення не впливає на тривалість поточного циклу, але такий цикл відповідатиме новому значенню *N*.

Діаграми роботи таймера, що ілюструють дію сигналу *GATE*, зображено на рис. 5.44, *а* для таймера КР580ВИ53 і таймера КР1810ВИ54 — на рис. 5.44, *б*. Для таймера КР580ВИ53 з появою *L*-рівня сигналу *GATE* лічення припиняється, а з появою *H*-рівня — відновлюється з перерваного значення. Для таймера КР1810ВИ54 з появою *L*-рівня сигналу *GATE* лічення також припиняється, а з появою *H*-рівня — починається зі значення коду попереднього встановлення.

Приклад 5.14. Запрограмувати лічильник *CT*0 у режим генератора прямокутних імпульсів для отримання частоти $f_{вих}$ = 1 кГц.

Візьмемо адреси таймера такі, як у прикладі 5.12. Для отримання послідовності імпульсів 1 кГц підключимо до виводу *G*0 сигнал *H*-рівня, а на вивід *CLK*0 — тактові імпульси з частотою 5 МГц.

Знаходимо значення коефіцієнта ділення:

$$N = \frac{f_{CLK}}{f_{вих}} = \frac{5000}{1} = 5000.$$

Згідно з рис. 5.43 визначимо керуюче слово для програмування лічильника *CT*0 у режимі 3, з двійково-десятковим засобом кодування:

$$00 \quad 11 \quad 011 \quad 1_2 \quad = 37H$$

Тоді програма буде така:

```
MOV   AL,37H      ;Програмування
OUT   16H,AL      ;таймера
MOV   AL,00       ;Запис молодшого
OUT   10H,AL      ;байта 00 попереднього встановлення
MOV   AL,50H      ;Запис старшого байта
OUT   10H,AL      ;попереднього встановлення
```

Після виконання програми на виводі *OUT*0 прямокутні імпульси з частотою 1 кГц триватимуть доти, доки не буде перепрограмовано таймер або вимкнено джерело живлення таймера.

5.5. Архітектура і функціональні можливості контролера прямого доступу до пам'яті

Контролер прямого доступу до пам'яті (КПДП) КР580ВТ57 призначений для організації швидкісного обміну даними між пам'яттю і зовнішніми пристроями, який ініціюється зовнішнім пристроєм (див. п. 5.1, рис. 5.5).

Структурну схему контролера зображено на рис. 5.45. Вона має:

- двоспрямований двостабільний буфер даних *BD*, призначений для обміну інформацією між МП і КПДП;
- схему керування читанням-записом *RWCU*, що адресує внутрішні регістри КПДП і керує обміном по шині $D7-D0$;
- блок керування *CU* містить регістри режиму і стану КПДП та задає режими роботи КПДП;
- блок керування пріоритетами *PCU* забезпечує порядок обслуговування запитів зовнішніх пристроїв;
- чотири канали прямого доступу $CH0-CH3$; кожний з них містить регістр адреси комірки пам'яті, де починається обмін, лічильник циклів обміну, два старших розряди якого відведені для задання операцій обміну та схему формування запитів-підтвердження.

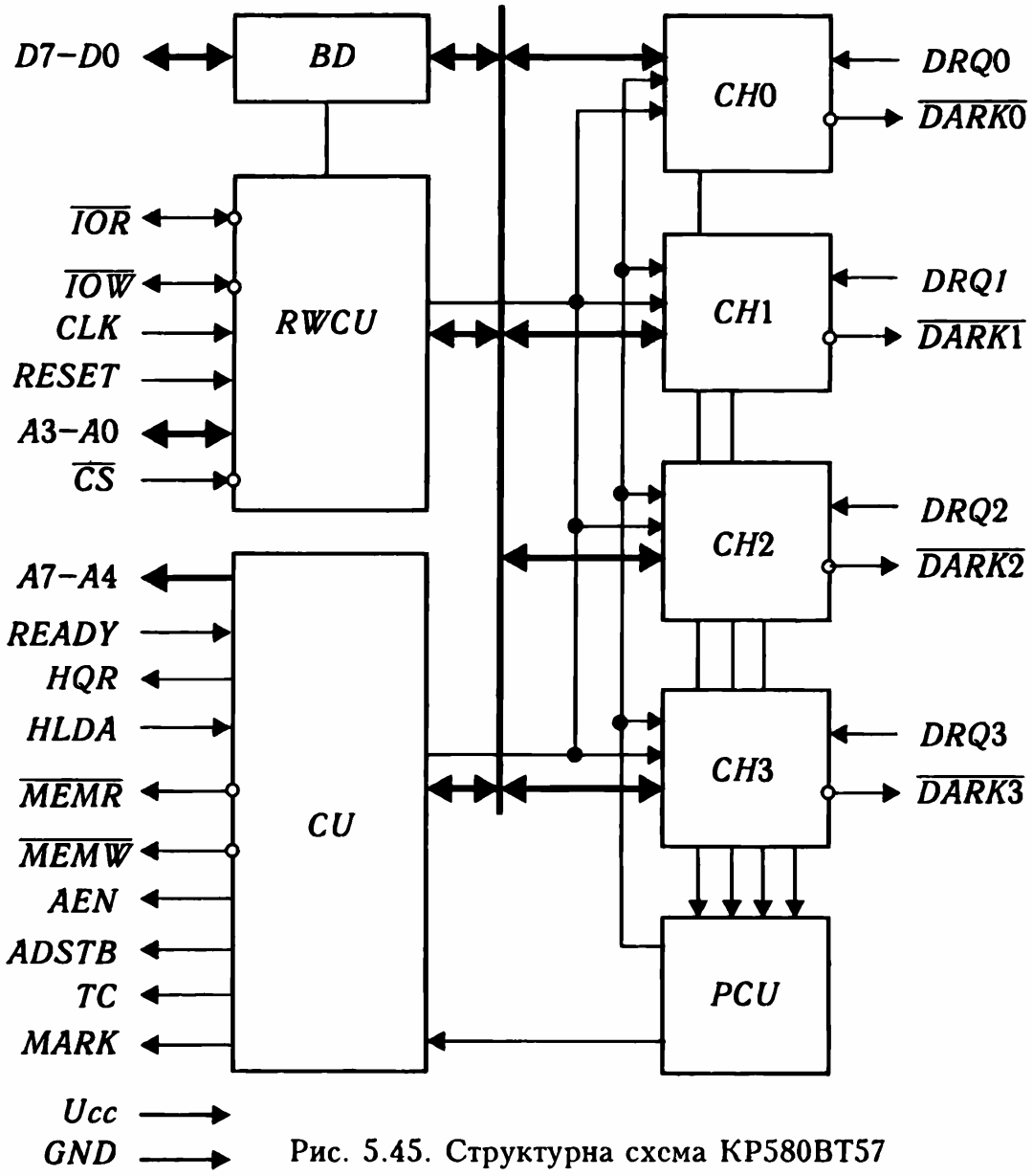


Рис. 5.45. Структурна схема КР580ВТ57

Призначення виводів *ВІС* наведено у табл. 5.11.

Кожний з чотирьох каналів ПДП забезпечує передавання блока даних ємністю до 16 Кбайт з довільною початковою адресою в діапазоні 0—64 Кбайт. Пріоритети каналів можуть бути фіксованими (канал 0 має найвищий пріоритет, канал 3 — найнижчий) або змінюватися циклічно. В останньому випадку каналу, в якому відбулося обслуговування запиту ПДП, присвоюється нижчий пріоритет, а каналу з наступним номером — вищий.

Схему підключення КПДП до системної шини зображено на рис. 5.46.

Молодший байт адреси пам'яті видається по лініях *A*3—*A*0 та *A*7—*A*4, які безпосередньо підключені до шини адреси *AB*. Старший байт адреси передається через шину *D*7—*D*0,

Таблиця 5.11. **Призначення виводів КПДП КР580ВТ57**

Позначення	Номер виводу	Призначення
D7—D0	21; 22; 23; 36; 27; 28; 29; 30	Входи-виходи даних для обміну з МП
$\overline{IOR}$	1	Читання введення-виведення — двонапрямлений тристабільний вхід-вихід. Вхідний сигнал *L*-рівня дозволяє зчитування інформації з КПДП; а вихідний сигнал низького рівня — з ПВВ
$\overline{IOW}$	2	Запис введення-виведення — двонапрямлений тристабільний вхід-вихід. Вхідний сигнал низького рівня дозволяє програмування КПДП, а вихідний сигнал низького рівня — запис у ПВВ
CLK	12	Вхід тактових імпульсів
RESET	13	Вхідний сигнал скидання
A3—A0	35; 34; 33; 32	Двонапрямлені тристабільні адресні виводи
$\overline{CS}$	11	Вибірка ВІС
A7—A4	40; 39; 38; 37	Тристабільні адресні виводи
READY	6	Вхідний сигнал готовності. *H*-рівень вказує на готовність КПДП до обміну
HOLD	10	Вихідний сигнал запиту захоплення. *H*-рівень вказує на запит захоплення КПДП системної шини
HLDA	7	Вхідний сигнал підтвердження захоплення. *H*-рівень вказує на дозвіл доступу до системної шини
$\overline{MEMR}$	3	Вихідний сигнал читання з пам'яті. Тристабільний вихід. *L*-рівень дозволяє читання комірки пам'яті, що адресується КПДП
$\overline{MEMW}$	4	Вихідний сигнал запису в пам'ять. Тристабільний вихід. *L*-рівень дозволяє запис у комірку пам'яті, що адресується КПДП
$\overline{AEN}$	9	Дозвіл адреси. *H*-рівень блокує шини адреси-даних
ASTB	8	Строб адреси. *H*-рівень вказує на знаходження на шині *D7—D0* старшого байта адреси ЗП

Продовж. табл. 5.11

Позначення	Номер виводу	Призначення
TC	36	Кінець лічення. *H*-рівень вказує на виконання останнього циклу передавання блоку даних
MARK	5	Маркер. *H*-рівень вказує, що до кінця передавання блока треба виконати кількість циклів обміну, що кратна 128
DRQ3–*DRQ0*	16; 17; 18; 19	Запити прямого доступу до пам'яті каналів *CH3*–*CH0*. *H*-рівснь вказує на запит від ПВВ
$\overline{DACK3}$–$\overline{DACK0}$	15; 14; 24; 25	Підтвердження запитів прямого доступу до пам'яті каналів *CH3*–*CH0*. *L*-рівень вказує на дозвіл обміну
Ucc	31	Напруга живлення +5 В
GND	20	Спільний вивід

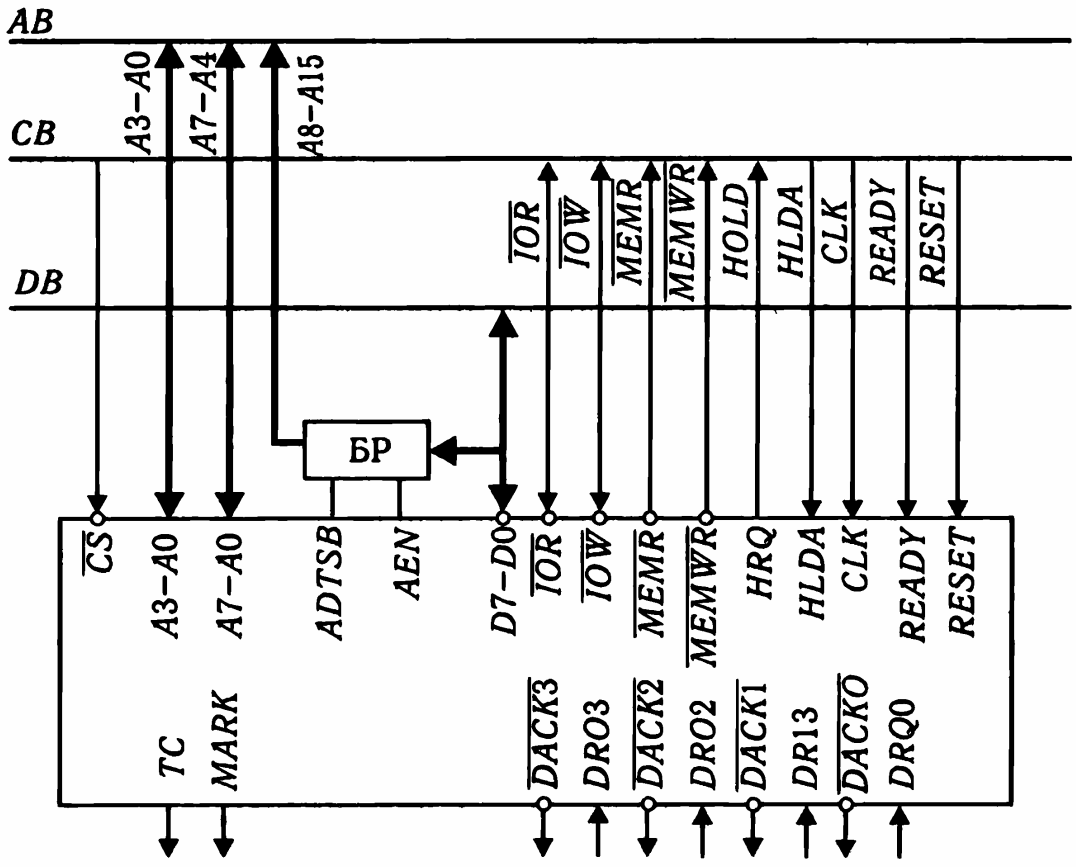


Рис. 5.46. Схема з'єднання КПДП із системною шиною з використанням буферного регістра

тому в схему підключено буферний регістр К589ИР12, який фіксує значення старшого байта за сигналом *ADSTB*, якщо *AEN* = 0. На вивід $\overline{CS}$ ВІС надходить сигнал з виходу дешифратора адрес введення-виведення. Інші виводи КПДП під'єднуються до однойменних ліній шин МП системи.

Під час ***програмування КПДП*** задають режим роботи каналів, напрям обміну інформацією між пам'яттю та пристроєм введення-виведення, початкову адресу та довжину масиву пам'яті.

Значення адресних розрядів *A*3—*A*0 і сигналу $\overline{CS}$ під час запису та читання регістрів ВІС наведено в табл. 5.12. Запис інформації у 16-розрядні регістри здійснюється двома командами, спочатку записується молодший байт, а потім — старший.

Режим роботи каналів задається керуючим словом, формат якого наведено на рис. 5.47. Розряди *D*3—*D*0 (*EN*3—*EN*0) дозволяють (за одиничного значення) або забороняють (за нуля) обмін у відповідному каналі.

Розряд *D*4 (*RP*) встановлює порядок обслуговування запитів від каналів. За *D*4 = 0 задається фіксований пріоритет каналів, а за *D*4 = 1 встановлюється режим циклічного пріоритету. Циклічний зсув пріоритетів відбувається після кожного циклу прямого доступу.

Встановлення розряду *D*5(*EW*) у стан логічної одиниці задає режим розширеного запису, за якого тривалість сигналів $\overline{IOW}$ і $\overline{MEMW}$, що генеруються КПДП, збільшуєть-

Таблиця 5.12. **Адресація внутрішніх регістрів КПДП**

Регістр	*CS*	*A*3	*A*2	*A*1	*A*0
*RQ*0	0	0	0	0	0
*CT*0	0	0	0	0	1
*RQ*1	0	0	0	1	0
*CT*1	0	0	0	1	1
*RQ*2	0	0	1	0	0
*CT*2	0	0	1	0	1
*RQ*3	0	0	1	1	0
*CT*3	0	0	1	1	1
Регістр режиму (запис)	0	1	0	0	0
Регістр стану (читання)	0	1	0	0	1
Відключення КПДП від шини даних	1	*x*	*x*	*x*	*x*

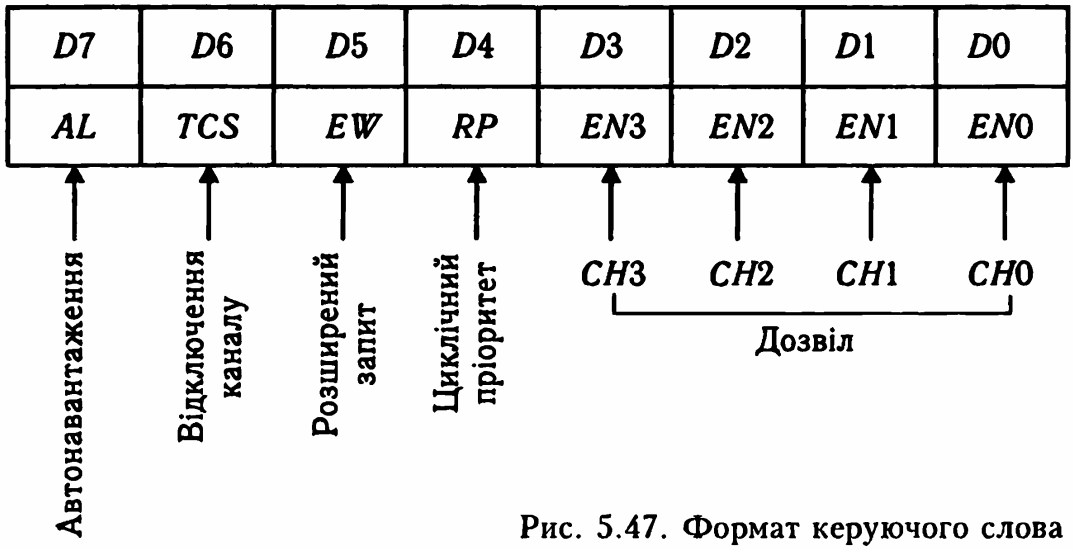


Рис. 5.47. Формат керуючого слова

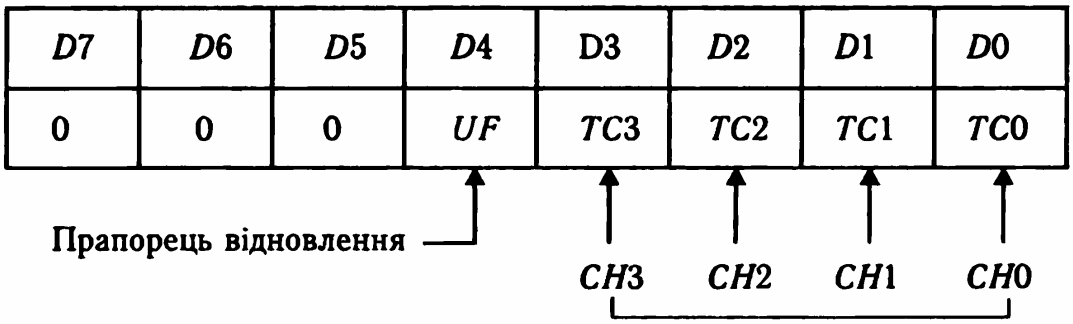


Рис. 5.48. Слово стану каналів

ся за рахунок зсуву переднього фронту. Це дозволяє ПВВ, що формує сигнал *READY* по фронту сигналу запису, зменшити або збільшити швидкість обміну.

За $D6(TCS) = 1$ поява сигналу *TC* в одному з каналів скидає відповідний розряд $D3-D0$, внаслідок чого канал від'єднується і його подальша робота можлива після перезавантаження регістра режиму. Якщо $D6(TCS) = 0$, то поява сигналу *TC* не впливає на розряд дозволу роботи каналу і закінчувати передавання має ПВВ припиненням вироблення сигналу *DRQ*.

Одиничне значення розряду $D7(AL)$ задає режим автозавантаження, в якому працює лише другий канал, використовуючи вміст своїх внутрішніх регістрів та внутрішніх регістрів третього каналу. Після передавання даних відповідно до параметрів регістрів другого каналу з появою сигналу *TC* вміст регістрів третього каналу автоматично завантажується у регістри другого каналу. При цьому в регістрі стану каналів (рис. 5.48) встановлюється у стан логічної одиниці розряд $D4(UF)$ — прапорець відновлення. Потім передавання даних триває відповідно до нових параметрів регістрів

другого каналу, а наприкінці першого циклу прямого доступу з новими параметрами прапорець відновлення скидається. Режим автозавантаження дає змогу організовувати повторні пересилання блоків даних з однаковими або з'єднувати кілька блоків з різними параметрами.

Розряди $D3-D0$ слова стану (див. рис. 5.48) встановлюються одночасно з появою сигналу TC відповідного каналу і скидаються сигналом $RESET$ під час читання вмісту регістра станів. Прапорець відновлення UF може бути скинутий, якщо записати логічний нуль у розряді $D7$ регістра режиму (див. рис. 5.47).

Початкова адреса ОЗП задається записом двох байтів у регістри каналів $RG0-RG3$.

Довжина масиву пам'яті та ***напрям обміну*** інформацією між пам'яттю і пристроєм введення-виведення задається записом двох байтів у лічильники $CT0$-$CT3$ циклів. Два старших розряди лічильника циклів визначають напрям обміну в такий спосіб: запис у пам'ять — 01, зчитування з пам'яті — 10, контроль — 00. Комбінація 11 є забороненою. Інші розряди лічильника визначають кількість байтів, що будуть переслані.

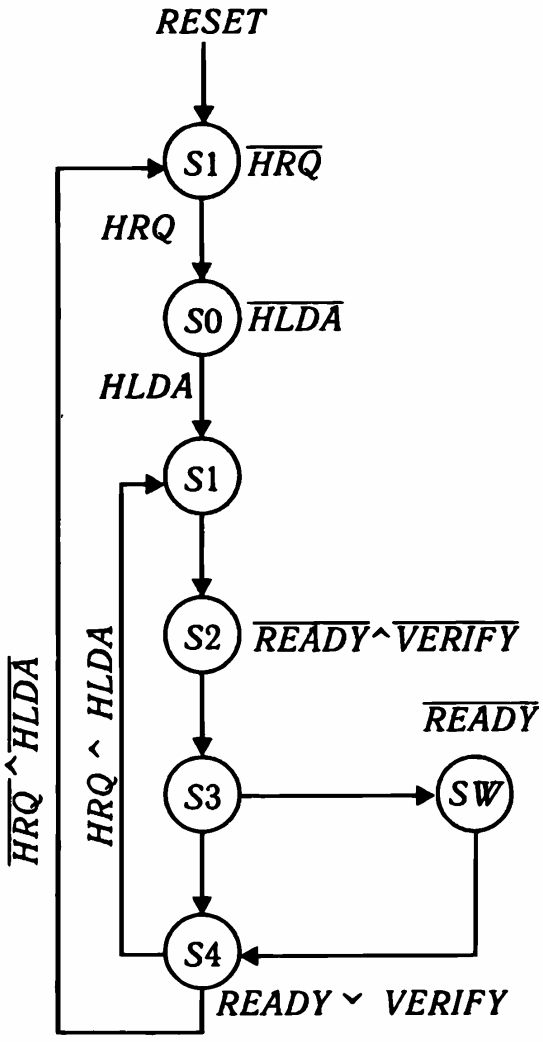


Рис. 5.49. Діаграма станів роботи КПДП

Якщо два старших розряди лічильника циклів каналів установлюють режим контролю $VERIFY$, то передавання даних не відбувається, оскільки не генеруються сигнали керування записом і читанням, усі інші функції прямого доступу зберігаються. Цей режим може використовуватися ПВВ для контролю прийнятих даних.

Роботу КПДП пояснює діаграма станів (рис. 5.49) та часові діаграми основних сигналів (рис. 5.50). Після запису слова режиму в регістр керуючого слова КПДП переходить у холостий стан SI, який триває доти, доки на

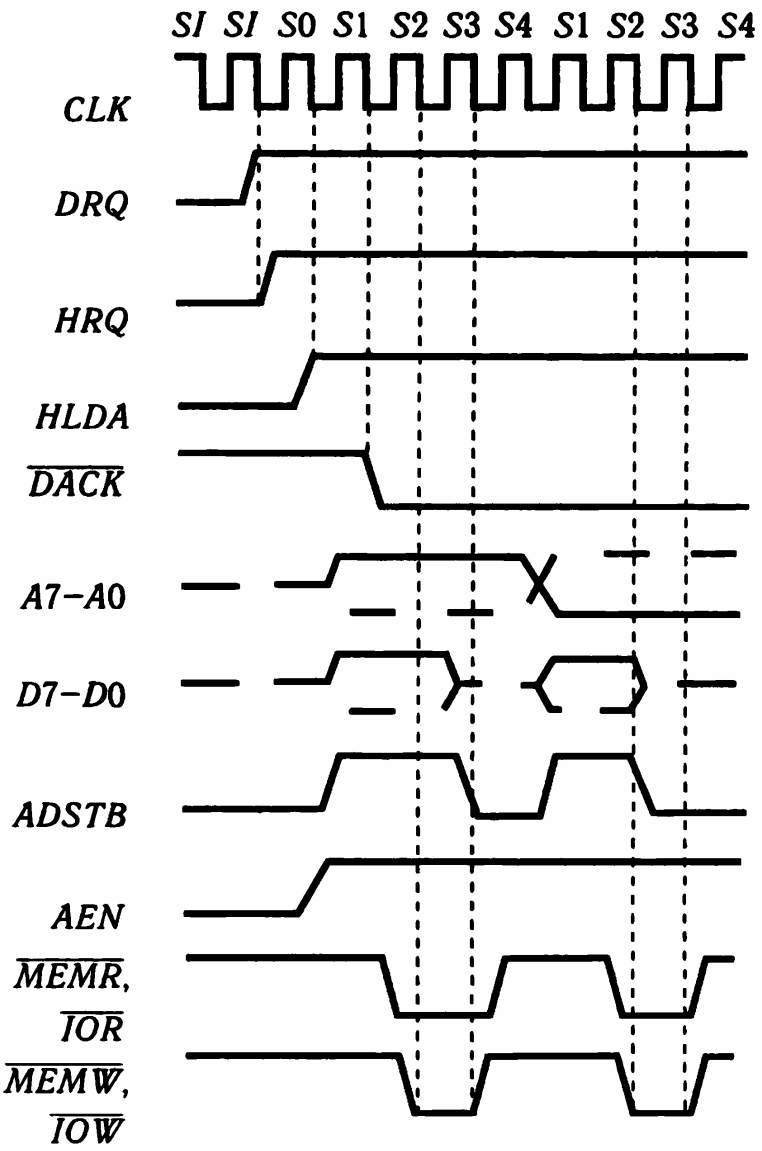


Рис. 5.50. Часові діаграми основних сигналів

один з входів КПДП не надійде запит *DRQ* від зовнішнього пристрою на прямий доступ до пам'яті. Переходячи у стан *S*0, він виробляє сигнал *HRQ* і очікує надходження від МП сигналу *HLDA*. Після надходження сигналу підтвердження *HLDA* починається цикл обміну. У стані *S*1 формується сигнал *AEN* для блокування інших пристроїв системи від шин даних і керування, видається код молодших розрядів на виходи $A7-A0$, а код старших розрядів — на виходи $D7-D0$. Видавання старших розрядів адреси супроводжується стробом *ADSTB* для запису їх у зовнішній буферний регістр. У стані *S*2 формуються сигнали $\overline{MEMR}$, $\overline{IOR}$ або $\overline{MEMW}$, $\overline{IOW}$, які визначають напрям обміну та сигнал $\overline{DACK}$, що вказує на початок обміну. У стані *S*3 здійснюється передавання даних у ЗП або ПВВ.

Стан *S4* завершує цикл прямого доступу. В цьому стані під час передавання останнього байта блока видається сигнал *TC*, а після закінчення — сигнал *MARK*. За потреби узгодження швидкодії ЗП і ПВВ за допомогою сигналу *READY* між станами *S3* і *S4* вводиться необхідна кількість станів очікування *SW*. У режимі контролю перехід у стан *SW* не дозволяється.

Приклад 5.15. Запрограмувати канал 1 КПДП на режим з фіксованими пріоритетами, з від'єднанням каналу після передавання, без автозавантаження і розширенного запису для пересилання по каналу 1 контролера 10 байт із ОЗП з початковою адресою 0000:0700*H* у пристрій введення. Вхід $\overline{CS}$ КПДП з'єднати з адресною лінією *A4*.

Визначимо адреси КПДП. Згідно з даними табл. 5.11, адреса регістра 1 каналу буде 02*H*, лічильника — 03*H*, регістра режиму — 08*H*.

Керуюче слово режиму визначимо відповідно до рис. 5.47.

Програма матиме такий вигляд:

```
MOV     AL,42H      ; Формування керуючого слова режиму
OUT     08H,AL      ; Запис його в регістр керуючого слова
MOV     AL,00       ; Молодший байт адреси ОЗП
OUT     02H,AL      ; переслати в регістр каналу 1
MOV     AL,07H      ; Старший байт адреси ОЗП
OUT     02H,AL      ; переслати в регістр каналу 1
MOV     AL,0AH      ; Молодший байт числа циклів
OUT     03H,AL      ; переслати в лічильник 1
MOV     AL,80H      ; Задати напрям передавання
OUT     03H,AL      ; читання пам'яті
```

Стан КПДП можна контролювати читанням вмісту *RGA*, *CT* і 8-розрядного регістра стану, спільного для всіх каналів, за допомогою команди *IN*. Для читання вмісту 16-розрядного регістра використовуються дві команди *IN* з тією самою адресною частиною, причому спочатку відбувається зчитування молодшого байта.

5.6. Програмований послідовний інтерфейс

Програмований послідовний інтерфейс КР580ВВ51 (*i*8251) — це універсальний синхронно-асинхронний приймач-передавач (УСАПП), призначений для організації обміну між МП і зовнішніми пристроями в послідовному форматі. Універсальний синхронно-асинхронний приймач-передавач приймає дані з 8-розрядної шини даних МП і передає їх у послідовному форматі периферійному пристрою або отримує послідовні дані від периферійного пристрою й перетворює їх на паралельну форму для передавання МП. Обмін може бути як *напівдуплексним* (односпрямованим), так і *дуплексним* (двонапрямленим). Послідовний інтерфейс здійснює обмін дани-

ми в асинхронному режимі зі швидкістю передавання до 9,6 Кбіт/с або в синхронному — зі швидкістю до 56 Кбіт/с залежно від запрограмованого режиму. Довжина переданих даних — від 5 до 8 біт. Під час передавання в МП символів завдовжки менше ніж 8 біт невикористані біти заповнюють нулями. Формат символу містить також службові біти і необов'язковий біт контролю парності.

Структурна схема УСАПП (рис. 5.51) містить:

- буфер передавача *TBF* зі схемою керування передавачем *TCU*, який призначений для приймання даних від МП і видавання їх у послідовному форматі на вихід *Tx*;
- буфер приймача *RBF* зі схемою керування приймачем *RCU*, що виконує приймання послідовних даних із входу *Rx* і передавання їх у МП у паралельному форматі;
- буфер даних *BD*, який є паралельним 8-розрядним двонапрямленим буфером шини даних з тристабільними каска-

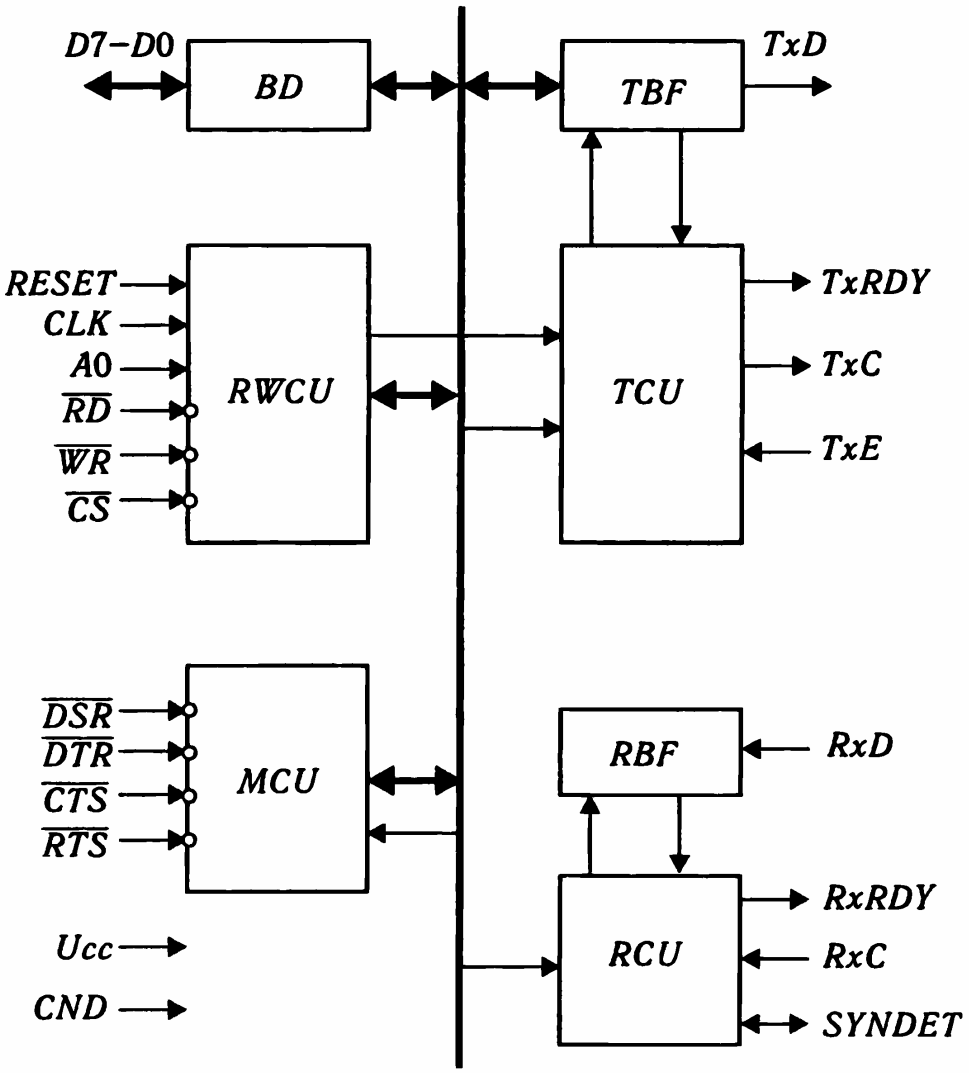


Рис. 5.51. Структурна схема УСАПП

дами, що використовується для обміну даними та керуючими словами між МП і УСАПП;

• блок керування читанням-записом *RWCU*, що приймає керуючі сигнали від МП і генерує внутрішні сигнали керування;

• блок керування модемом *MCU*, що обробляє керуючі сигнали, призначені для зовнішнього пристрою.

Призначення виводів ВІС УСАПП наведено у табл. 5.13.

З'єднання УСАПП із шинами МП показано на рис. 5.52.

Таблиця 5.13. **Призначення виводів УСАПП**

Позначення виводу	Номер виводу	Призначення
D7—D0	8; 7; 6; 5; 2; 1; 28; 27	Канал даних
RESET	21	Установлення «0» (вихідний стан)
CLK	20	Синхронізація
A0	12	Низький рівень визначає можливість запису або читання даних в(із) УСАПП; високий рівень визначає можливість запису керуючих слів або читання слова стану в (із) УСАПП
$\overline{RD}$	13	Читання-дозвіл виведення даних або слова стану з УСАПП на шину даних
$\overline{WR}$	10	Запис-дозвіл введення інформації із шини даних
$\overline{CS}$	11	Вибірка кристала-з'єднання УСАПП із шинами даних МП
$\overline{DSR}$	22	Готовність передавача термінала
$\overline{DTR}$	24	Запит передавача термінала
$\overline{CTS}$	17	Готовність приймача термінала
$\overline{RTS}$	23	Запит приймача термінала
SYNDET	16	Вид синхронізації: для синхронного режиму вихідний сигнал високого рівня — ознака внутрішньої синхронізації; для синхронного режиму із зовнішньою синхронізацією сигнал є вхідним; в асинхронному режимі сигнал є вихідним
RxC	25	Синхронізація приймача
RxRDY	14	Готовність приймача
RxD	3	Вхід приймача
TxC	9	Синхронізація передавача

Продовж. табл. 5.13

Позначення виводу	Номер виводу	Призначення
$\overline{TxE}$	18	Кінець передавання. Сигнал високого рівня є ознакою закінчення посилання даних
TxRDY	15	Готовність передавача
TxD	19	Вихід передавача
U_{CC}	26	Напруга живлення +5 В

Сигнал *Ai*, поданий на вивід ВІС *A*0, визначає дві адреси УСАПП. Для адреси з *Ai* = 0 передаватимуться дані, а за *Ai* = 1 записуватимуться команди або читатиметься слово стану. Інші виводи під'єднані до однойменних ліній шин МП системи.

Програмування УСАПП відбувається завантаженням керуючих слів. Розрізняють керуючі слова двох типів: керуюче слово ініціалізації та операційне керуюче слово.

Значення сигналів адреси *A*0, керування читанням $\overline{RD}$, записом $\overline{WR}$ і вибіркою $\overline{CS}$ під час запису та читання регістрів ВІС подано у табл. 5.14.

Керуюче слово ініціалізації задає синхронний або асинхронний режим роботи, формат даних, швидкість приймання або передавання, контроль правильності даних. Це слово заноситься відразу після встановлення УСАПП у вихідний стан відповідною програмою або за сигналом *RESET*, а замінюється лише у разі зміни режиму. Формат керуючого слова різний в асинхронному чи синхронному режимах.

В асинхронному режимі роботи дані, що передаються, містять нульовий старт-біт, біти даних, біт контролю і стоп-біти. Кількість бітів даних і стоп-бітів та наявність або відсутність біта контролю задаються записом в УСАПП керую-

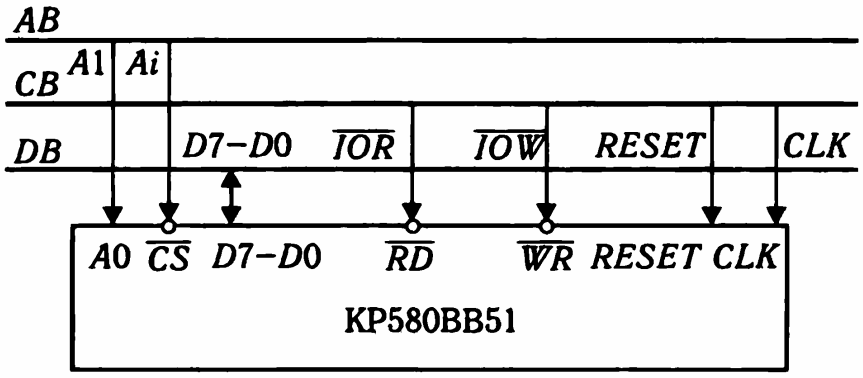


Рис. 5.52. З'єднання УСАПП із шинами МП

Таблиця 5.14. **Визначення операцій сигналами керування від МП**

Операція	Сигнали керування			
	*A*0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$
Читання даних з УСАПП на шину *D*7 – *D*0	0	0	1	0
Запис даних із шини *D*7 – *D*0 в УСАПП	0	1	0	0
Зчитування слов стану з УСАПП на шину *D*7 – *D*0	1	0	1	0
Запис керуючого слова із шини *D*7 – *D*0 в УСАПП	1	1	0	0
Високоімпедансний стан виводів	*x*	1	1	0
*D*7 – *D*0	*x*	*x*	*x*	1

чого слова режиму (рис 5.53, *а*). Розряди *D*0 і *D*1 визначають коефіцієнт ділення сигналів синхронізації *CLK*. Розряди *D*3 і *D*2 визначають кількість бітів даних. Режим контролю задається розрядами *D*5 і *D*4; за *D*4 = 0 контроль парності заборонений; значення розряду *D*5 встановлює вид контролю — парності або непарності. Розряди *D*7 і *D*6 визначають кількість переданих стоп-бітів.

Синхронний обмін передбачає передавання даних у вигляді масивів слів. Для синхронізації запуску під час приймання даних використовуються один або два символи синхронізації (спеціальні кодові комбінації, наприклад 10010100). Формат керуючого слова ініціалізації режиму для синхронного обміну наведено на рис. 5.53, *б*. Розряди *D*1 і *D*0 мають нульове значення. Розряд *D*6 встановлює тип синхронізації (зовнішню або внутрішню). Розряд *D*7 визначає використання одного (*D*7 = 1) або двох (*D*7 = 0) символів синхронізації. Призначення розрядів *D*3, *D*2 і *D*5, *D*4 аналогічне призначенню цих розрядів під час асинхронного обміну.

Контроль стану УСАПП у процесі обміну даними МП здійснюється за допомогою команди читання слова стану. На рис. 5.53, *в* показано формат слова стану УСАПП. Розряд *D*3(*PE*) установлюється тоді, коли виникають помилки парності; розряд *D*4(*OE*) — коли виникають помилки переповнення, якщо МП не прочитав символ; розряд *D*5 — коли виникає помилка, внаслідок того, що для асинхронного режиму не виявлено стоп-біт. Інші розряди слова стану мають таке саме значення, що й однойменні виводи МП, які наведені в табл. 5.13.

Керування роботою УСАПП після ініціалізації здійснюється записуванням *операційних керуючих слів*, які можуть ба-

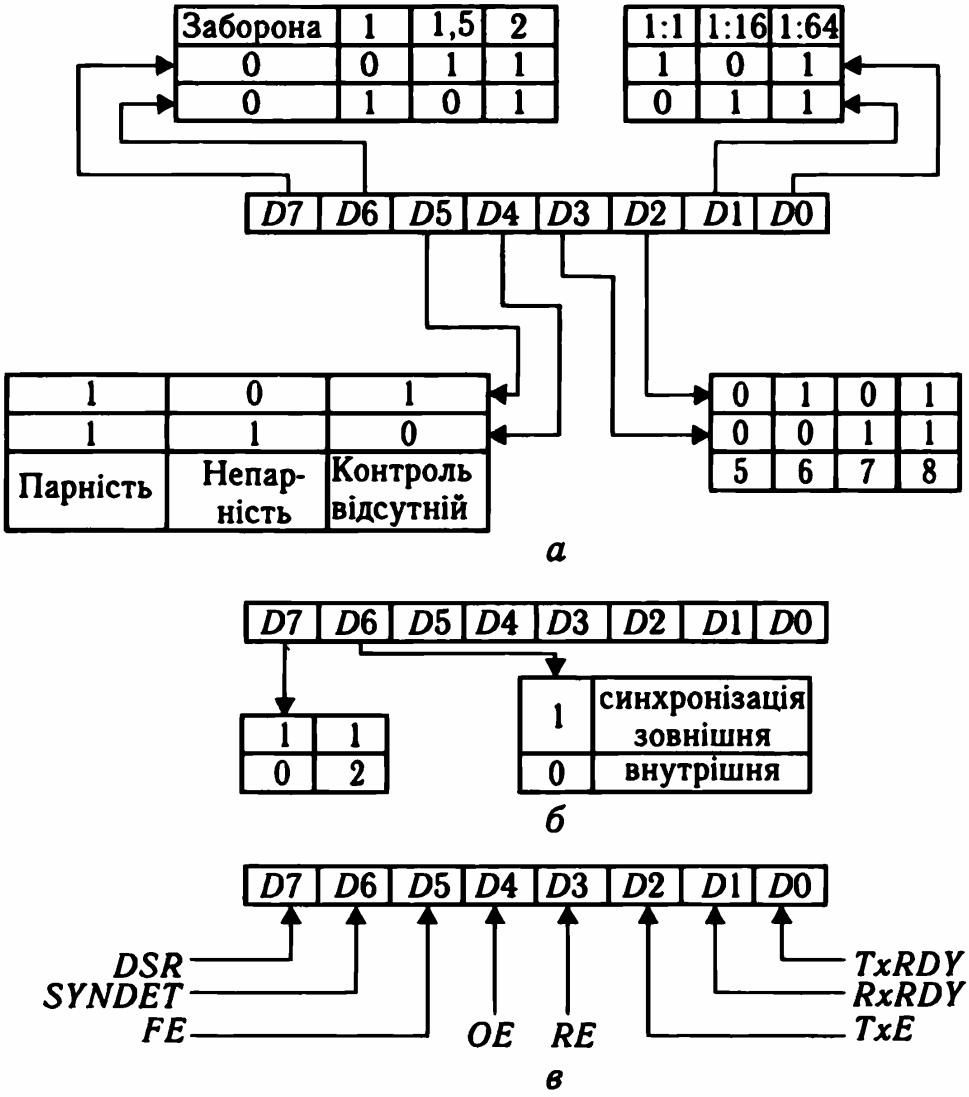


Рис. 5.53. Формат керуючого слова ініціалізації:
а — для асинхронного обміну; *б* — для синхронного обміну; *в* — формат слова стану

гаторазово задаватися у процесі обміну, керуючи різними його етапами. Призначення окремих розрядів операційного керуючого слова УСАПП наведено в табл. 5.15.

Під час асинхронного обміну операційне керуюче слово завантажується відразу після керуючого слова ініціалізації, а під час синхронного — перед ним розміщуються один або два символи синхронізації.

Часові діаграми сигналів керування УСАПП під час запису керуючого слова режиму, символів синхронізації команди наведено на рис. 5.54, *а*, а часові діаграми сигналів під час читання слова стану — на рис. 5.54, *б*.

Запис керуючих слів і символів синхронізації здійснюється через шину даних *DB* у процесі подання на вхід $C/\overline{D}$ *H*-рівня, а на вхід $\overline{WR}$ — *L*-рівня (див. рис. 5.54, *а*). Після

Таблиця 5.15. **Призначення розрядів операційного керуючого слова УСАПП**

Розряд	Позначення	Призначення
*D*0	*TxE*	Дозвіл передавання. За нульового значення передавання інформації неможлива, за одиничного — можлива
*D*1	*DTR*	Запит про готовність передавача до передавання. За одиничного значення — запис нуля на виході *DTR*
*D*2	*RxE*	Дозвіл на прийом. За нульового значення приймання інформації неможливе, за одиничного — можливе
*D*3	*SBRK*	Кінець передавання. За нульового значення — нормальна робота каналу передавання, при одиничному значенні — встановлення високого рівня на виводі *TxD*
*D*4	*ER*	Виявлення помилок. За одиничного значення — встановлення розрядів помилок у вихідний стан
*D*5	*RTS*	Запит про готовність приймача термінала до приймання. Запис нуля на вході *RTS* за *D*5 = 1
*D*6	*IR*	Програмне скидання схеми у вихідний стан. При одиничному значенні — встановлення УСАПП у вихідний стан і готовність до приймання інструкції режиму
*D*7	*EH*	Режим пошуку імпульсів синхронізації. За одиничного значення — встановлення режиму пошуку символів синхронізації

початкового встановлення УСАПП приймає інформацію на *DB* як керуюче слово ініціалізації і розміщує його у відповідному регістрі. Блок *RWCU* дешифрує це слово і, якщо запрограмований асинхронний режим, то наступне слово сприймається як операційне керуюче слово, а якщо синхронний — інформація на *DB* сприймається як символ синхронізації.

Читання слова стану (див. рис. 5.54, *б*) здійснюється під час подання на вхід $C/\overline{D}$ *H*-рівня, а на $\overline{RD}$ — *L*-рівня. Інформацію, яка міститься у слові стану (див. рис. 5.53, *в*), можна використовувати для організації обміну між МП і УСАПП, наприклад, визначати, чи не було помилки під час передавання інформації, який стан готовності приймача (передавача) до обміну.

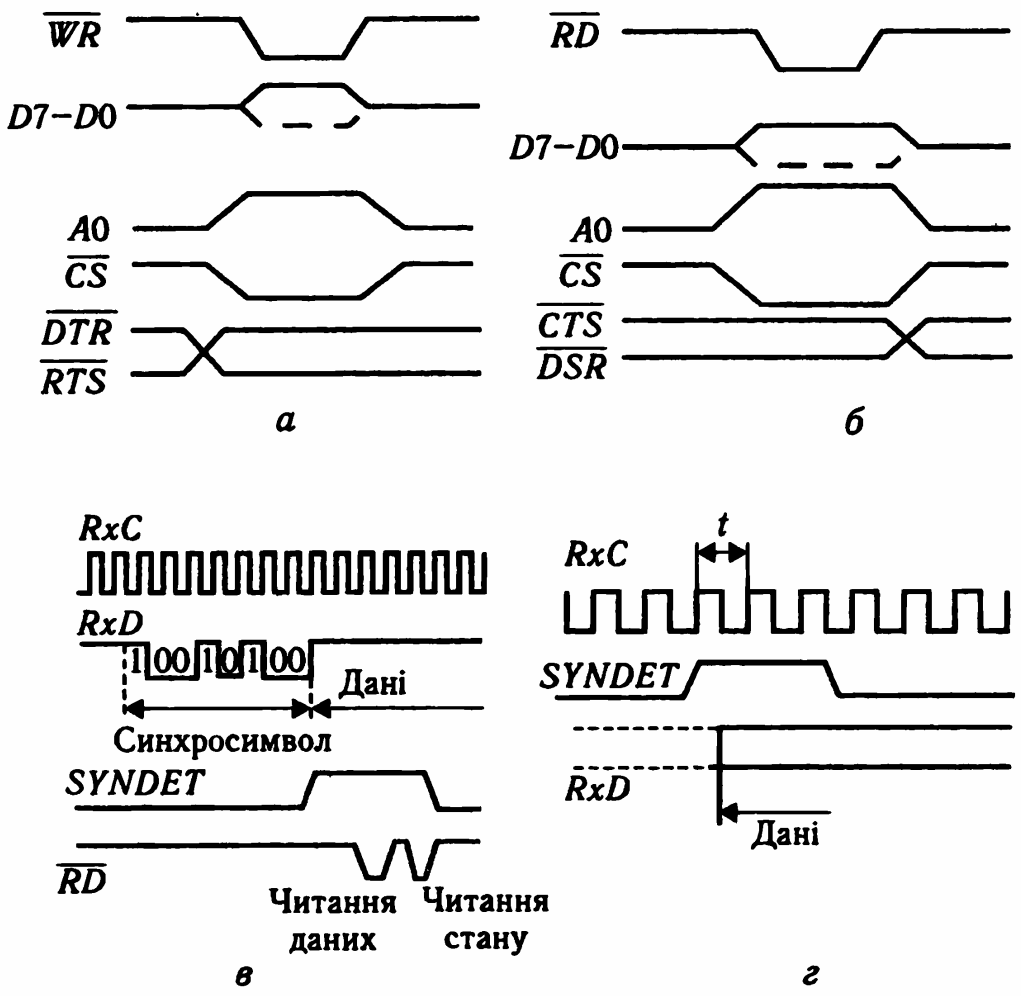


Рис. 5.54. Часові діаграми сигналів керування УСАПП:
а — запис керуючого слова режиму, символів синхронізації та операційного слова; *б* — читання слова стану; *в* — приймання вхідних даних з внутрішньою синхронізацією; *г* — синхронне приймання із зовнішньою синхронізацією

Після запису керуючого слова режиму та операційного слова УСАПП готовий до виконання обміну даними в одному з п'яти режимів:

- синхронне передавання;
- синхронне приймання з внутрішньою синхронізацією;
- синхронне приймання із зовнішньою синхронізацією;
- асинхронне передавання;
- асинхронне приймання.

Під час *синхронного передавання даних* на виході *TxD* з частотою сигналу синхронізації формується послідовність, що починається з символів синхронізації, запрограмованих керуючим словом режиму (рис. 5.53, *б*). Потім передаються коди символів, що надходять з МП, кожний з яких може закінчуватися бітом контролю. Якщо МП не завантажив чер-

говий символ до моменту передавання, то УСАПП вставляє в передану послідовність символи синхронізації, а на виході *TxD* генерується сигнал високого рівня, який вказує на порожній регістр передавання.

У режимі *синхронного приймання з внутрішньою синхронізацією* (див. рис. 5.54, *в*) УСАПП починає роботу з пошуку у вхідній послідовності символів синхронізації. УСАПП порівнює записані в нього під час програмування символи синхронізації з прийнятими символами. Після виявлення символів синхронізації на виводі *SYNDET* встановлюється сигнал високого рівня і починається приймання вхідних даних (див. рис. 5.54, *в*). Сигнал на виводі *SYNDET* автоматично скидається у процесі читання слова стану УСАПП.

Під час *синхронного приймання із зовнішньою синхронізацією* (див. рис. 5.54, *г*) на вивід *SYNDET* подається сигнал від зовнішнього пристрою, що дозволяє приймання даних на вході *RxD* зі швидкістю сигналів синхронізації, які надходять на вхід *RxD*. Можлива організація приймання даних у МП за перериванням, якщо сигнали на виводі *SYNDET* використовуються як запит переривання.

У режимі *асинхронного передавання* послідовні дані формуються на виході *TxD* за заднім фронтом сигналу синхронізації *TxC* з періодом, що задається керуючим словом режиму і дорівнює 1, 16 або 64 періодам сигналу синхронізації. Якщо після передавання символу наступного символу немає, то на виході *TxD* встановлюється напруга високого рівня доти, доки нові дані не надійдуть від МП. У програмі, яка реалізує алгоритм асинхронного передавання, запис чергового байта в УСАПП здійснюється за командою виведення *OUT*, якщо у слові стану розряд *D*0 = 1. Сигнал на виході *TxDY* використовується як сигнал запиту переривання.

Асинхронне приймання даних починається з пошуку старт-біта, який встановлює на вході *RxD* напруга *L*-рівня. Наявність цього біта вдруге перевіряється внутрішнім строб-імпульсом. Якщо старт-біт підтверджений, то запускається внутрішній лічильник бітів, який визначає початок і кінець бітів даних, біт контролю і стоп-біт. Приймання стоп-біта вказує на закінчення приймання байта інформації і супроводжується встановленням сигналу високого рівня на виході *RxRDY*. У програмі асинхронного приймання передавання чергового байта даних у МП може здійснюватися за командою введення *IN*, якщо в слові стану розряд *D*1 = 1, що відповідає *H*-рівню сигналу на виході *RxRDY*, або за перериванням, якщо сигнал на виході *RxRDY* використовується як сигнал запиту переривання.

Приклад 5.16. Запрограмувати УСАПП на асинхронний режим з наступним форматом даних — старт-біт, 8 біт даних, біт контролю парності та 1,5 стоп-біта. Коефіцієнт ділення частоти синхросигналів 1/64. Вхід $\overline{CS}$ УСАПП з'єднати з адресною лінією $A4$, вхід $A0$ з лінією $A1$. Здійснити передавання даних із регістра BL по послідовному порту.

Визначимо адреси УСАПП. Запис керуючого слова відбуватиметься, якщо $A4 = 0$, $A1 = 1$, тобто за адресою 02, а передавання даних — якщо $A1 = 0$, тобто за адресою 00.

Згідно з рис. 5.53, *а* керуюче слово ініціалізації дорівнює 10111111.

Операційне керуюче слово визначимо за табл. 5.14 як 00010001. Одиничне значення біта $D0$ дозволяє передавання, біт $D4$ скидає значення розрядів помилок у слові стану у вихідне положення. Нульове значення біта $D3$ визначає нормальну роботу каналу передавання. Інші біти не впливають на цей режим.

```
        MOV AL,10111111B  ; Формування в AL керуючого слова
                          ; ініціалізації
        OUT 02,AL         ; і пересилання його в УСАПП
        MOV AL,00010001B  ; Формування в AL операційного ке-
                          ; руючого слова
        OUT 02,AL         ; і пересилання його в УСАПП
M1:     IN AL,02          ; Читання слова стану
        AND AL,01         ; Виділення розряду D0
        JZ M1             ; Якщо D0 = 0 (УСАПП не готовий до
                          ; передачі),
                          ; то відбувається перехід на M1
        MOV AL,BL
        OUT 00,AL         ; Пересилання даних
```

Ознакою того, що дані можна передавати по послідовному порту, тобто УСАПП готовий до обміну, в цій програмі прийнята умова $D0 = 1$ у слові стану. Іншим шляхом є використання сигналу на виході $TxDY$ як запиту переривання.

5.7. Програмований контролер переривань

Програмований контролер переривань (ПКП) КР580ВН59А — пристрій, що реалізує в МПС оброблення запитів переривань від зовнішніх пристроїв, наприклад, датчиків аварійних ситуацій або пристроїв введення-виведення, що реалізують протокол обміну за перериванням (див. п. 5.1). ВІС ПКП виконує такі функції:

- запам'ятовує запити переривання, які задаються переднім фронтом або потенціалом;
- маскує, тобто забороняє виконання обраних запитів;
- формує вектор переривання і виконує дії щодо переходу на підпрограму оброблення запиту;
- формує сигнал переривання для МП;
- виконує пріоритетне оброблення запитів переривання.

ВІС КР580ВН59А залежно від того, як її запрограмували, може виробляти або код команди 8-розрядного МП *i*8080 *CALL ADRV*, де *ADRV* — адреса підпрограми оброблення, або видавати на шину даних номер переривання *n* для команди *INT n* 16-розрядного МП *i*8086.

Одна ВІС ПКП обробляє вісім запитів на переривання, але за каскадного вмикання ВІС кількість запитів переривання може бути збільшена до 64. Спрощену структурну схему ПКП зображено на рис. 5.55. До складу програмованого контролера переривань входять: двонапрямлений 8-розрядний буфер даних (*BD*), призначений для з'єднання ПКП із системною інформаційною шиною; блок керування читанням-записом (*RWCU*), що приймає керуючі сигнали від МП і задає режим функціонування ПКП; схема каскадного буфера — компаратора (*CMP*), яка використовується під час вмикання в систему кількох ПКП; схема керування (*CU*), що формує сигнали переривання і трибайтову команду *CALL* або вектор переривання *n*; регістр запитів переривань (*RGI*), що використовується для зберігання всіх запитів переривань; схема оброблення за пріоритетами (*PRB*), яка ідентифікує пріоритети запитів і вибирає запит з найвищим пріоритетом; регістр обслуговуваних переривань (*ISR*), що зберігає рівні запитів переривань, які знаходяться на обслуговуванні ПКП; регістр маскування переривань (*RGM*), що забезпечує заборону однієї або кількох ліній запитів переривання.

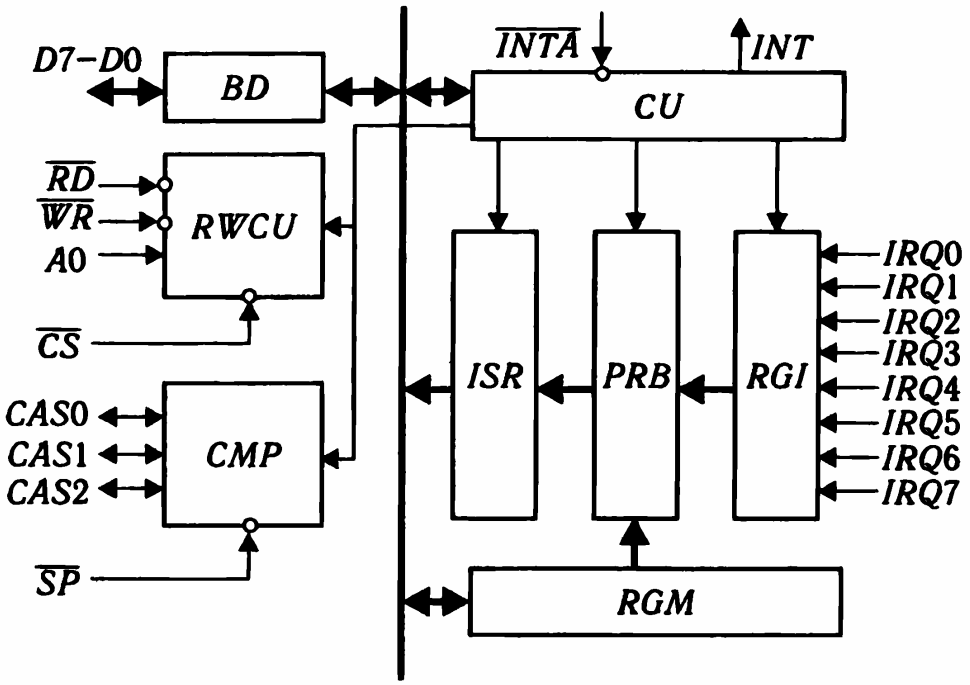


Рис. 5.55. Структурна схема ПКП

Таблиця 5.16. **Опис виводів ПКП**

Позначення виводу	Номер контакта	Призначення виводу
D7—D0	4; 5; 6; 7; 8; 9; 10; 11	Вхід-вихід даних
$\overline{RD}$	3	Вхід стробу читання
$\overline{WR}$	2	Вхід стробу запису
A0	27	Вхід нульового розряду адреси, що використовується під час завантаження команд і зчитування стану ПКП
$\overline{CS}$	1	Вхід вибору мікросхеми
CAS0-CAS2	12; 13; 15	Входи-виходи каскадування
SP	16	Ознака підпорядкування: напруга *H*-рівня вказує, що ПКП є старшим (головним) контролером; напруга *L*-рівня визначає ПКП підпорядкованим (відомим) контролером
INTA	26	Підтвердження переривання — вхідна напруга *H*-рівня вказує про видавання ПКП команди *CALL* на шину
INT	17	Переривання — напруга *H*-рівня вказує про запит на обслуговування
IRQ0-IRQ7	18; 19; 20; 21; 22; 23; 24; 25	Входи запитів переривань (передній фронт)
U_{CC}	28	Напруга живлення (+5 В)
GND	14	Спільний вивід (0 В)

Призначення вхідних, вихідних і керуючих сигналів ПКП наведено в табл. 5.16 під час опису виводів мікросхеми.

З'єднання ВІС КР580ВН59 зі стандартною системною шиною показано на рис. 5.56.

Схема (див. рис. 5.55) працює так. Запити переривань від зовнішніх пристроїв надходять на входи *IRQ0—IRQ7* і запам'ятовуються в регістрі *RGI*. Після цього сигнали надходять на схему оброблення за пріоритетами *PRB*, дозволяє або не дозволяє подальшому проходженню запиту переривання залежно від його пріоритету та заборони маскуванням. Будь-який запит переривання можна заборонити після запису маски в регістр *RGM*. Потім дозволені запити надходять у регістр *ISR* і встановлюють відповідні розряди. Схема керування (*CU*) на основі стану регістра *ISR* формує сигнал переривання *INT* для МП. Мікропроцесор приймає сигнал *INT* і, якщо переривання дозволені, підтверджує приймання вида-

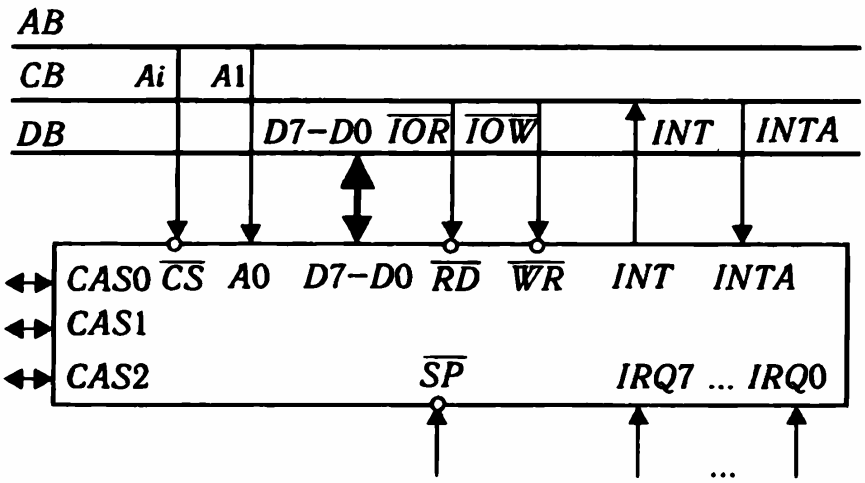


Рис. 5.56. З'єднання ВІС КР580ВН59 зі стандартною системною шиною

чею сигналу *INTA*. Після отримання сигналу *INTA* ПКП видає на шину $D7-D0$ код команди *CALL* або вектор переривання *n*. У першому випадку МП видає ще два сигнали *INTA*, що дозволяють ПКП передати на шину даних 16-розрядну адресу підпрограми обслуговування переривання, причому молодший байт адреси передається за першим, а старший — за другим *INTA* сигналом. У другому МП видає ще один сигнал *INTA*, за допомогою якого МП зчитує значення вектора переривання *n*.

Програмований контролер може працювати і в режимі опитування запитів переривання. В цьому разі МП зчитує код запиту з найвищим рівнем пріоритету за сигналом $\overline{RD}$. Приймання запитів, маскування та аналіз пріоритету виконуються так само, як і під час обслуговування переривань за запитом.

Для збільшення кількості рівнів переривання ПКП можуть бути об'єднані у систему, що складається з одного ведучого і кількох ведених ПКП (рис. 5.57).

У процесі обслуговування запиту, що надійшов на вхід веденого ПКП, ведучий ПКП за сигналом *INTA* видає на шину даних код команди *CALL*, а на шини $CAS0-CAS2$ — код номера веденого ПКП. З надходженням другого і третього сигналів *INTA* адреса підпрограми обслуговування даних видає обраний за кодом на шинах $CAS0-CAS2$ ведений ПКП. У разі використання ПКП в МПС з *i*8086 ведений ПКП видає за другим сигналом *INTA* значення номера переривання *n*.

Програмування ПКП полягає у запису в нього у визначеному порядку від 2 до 4 керуючих слів ініціалізації (*ICW*). Далі в будь-якому порядку можна записувати керуючі операційні слова (*OCW*) залежно від необхідних функцій ПКП.

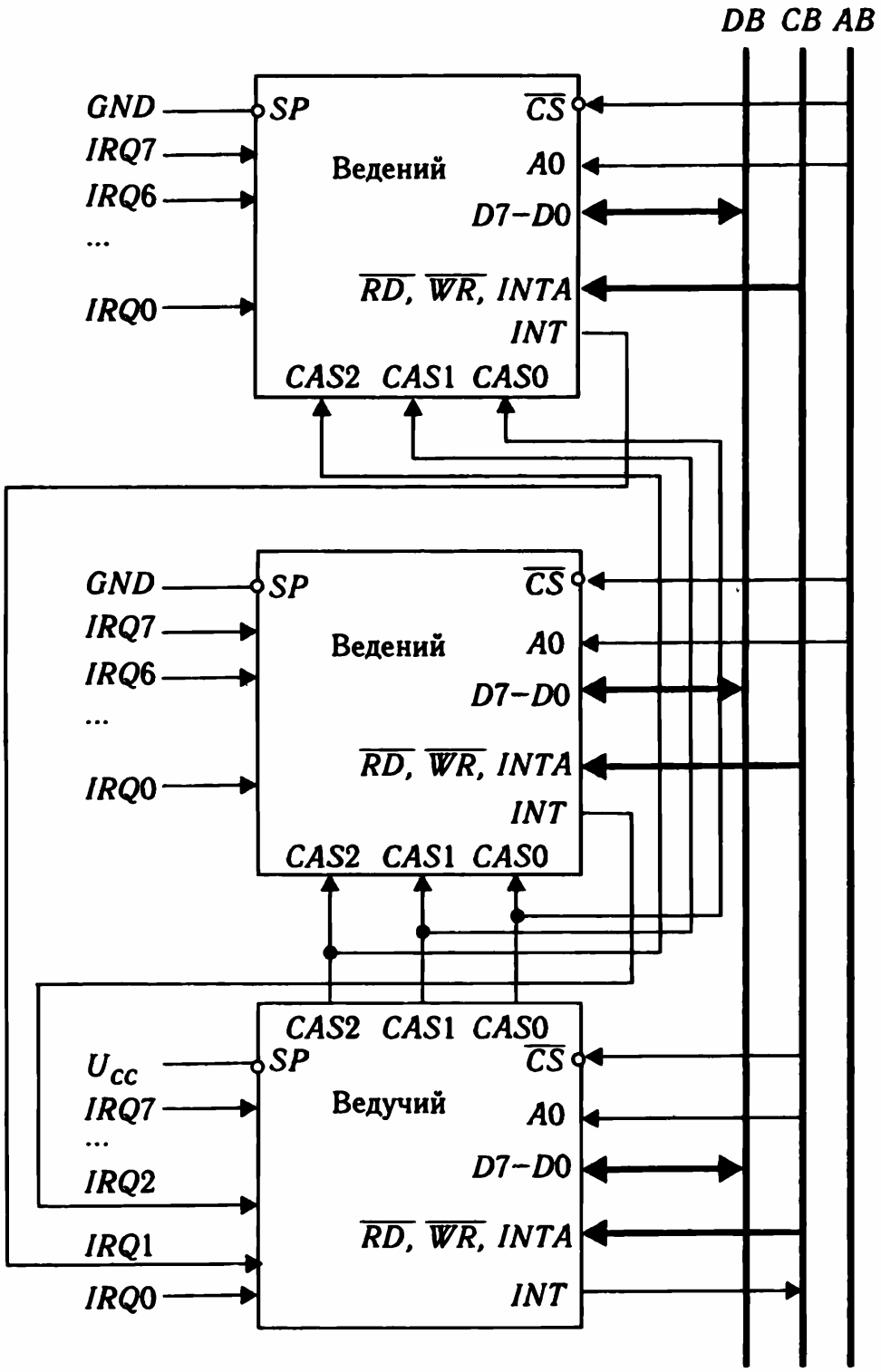


Рис. 5.57. Каскадне з'єднання ПКП

Керуюче слово *ICW*1 (рис. 5.58, *а*) скидає регістри *RGI*, *RGM* і присвоює нижчий пріоритет входу *IRQ*7. Одиничне значення розряду *AD* (*D*0) вказує на те, що треба використати додатне керуюче слово ініціалізації *ICW*4.

Розряд *S* цього слова визначає наявність одного або кількох ПКП у системі. Інші розряди можуть приймати будь-яке

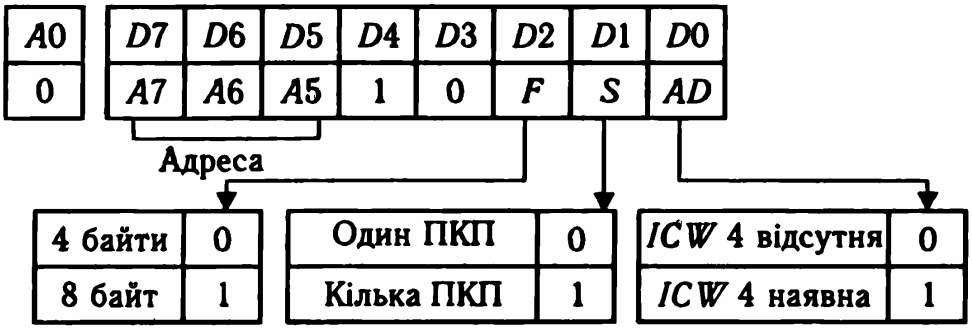


Рис. 5.58. Формат керуючого слова ініціалізації *ICW*1

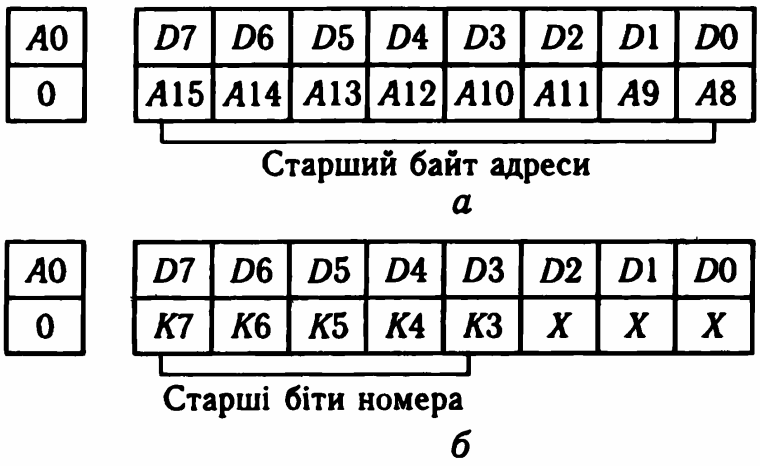


Рис. 5.59. Формати керуючих слів ініціалізації *ICW*2:
а — для системи з МП *i*8080; *б* — з *i*8086

значення в МПС з *i*8086. В МПС з 8080 розряд *F* (формат) визначає адресний інтервал 4 або 8 байт між початковими адресами підпрограм обслуговування переривань. Розряди *A*7—*A*5 керуючого слова *ICW*1 використовуються для формування молодшого байта адрес підпрограм обслуговування переривань відповідно до табл. 5.16.

Керуюче слово *ICW*2 має різний вигляд для МП 8080 і 8086. У першому випадку (рис. 5.59, *а*) воно є старшим байтом адреси підпрограми обслуговування переривань, що видається ПКП на шину даних як третій байт команди *CALL*.

У другому випадку (див. рис. 5.59, *б*) розряди *D*7—*D*3 визначають старші біти номера переривання для кожного входу *IRQ*. Молодші три біта визначаються номером входу *IRQ*, на який надійшов запит переривання (табл. 5.17).

У мікропроцесорній системі, що складається з кількох ПКП, для кожного з них після двох перших керуючих слів ініціалізації вводиться слово *ICW*3, що задає режим взаємодії контролерів. У керуючому слові *ICW*3 для ведучого ПКП (рис. 5.60, *а*) наявність логічної одиниці в одному з розрядів

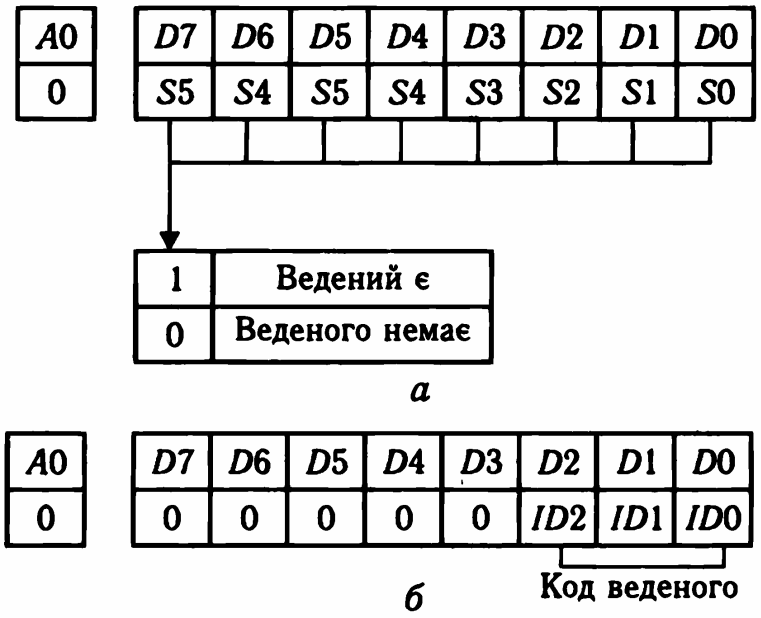


Рис. 5.60. Формати керуючих слів ініціалізації *ICW*3:
а — для ведучого ПКП; *б* — для веденого

*A*0	*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
1	0	0	0	*P*	*SP*	*MIS*	*E*	8/16

Рис. 5.61. Формат керуючого слова ініціалізації *ICW*4

вказує на під'єднання до відповідного входу запиту переривання виходу *INT* веденого ПКП. У керуючому слові *ICW*3 для веденого ПКП (рис. 5.60, *б*) задається код його номера у системі.

Формат керуючого слова ініціалізації *ICW*4 наведено на рис. 5.61. На рис. 5.58—5.61 ліворуч наведено значення входу *A*0, за якого треба завантажувати керуючі слова.

Нульове значення розряду *P*(*D*4) (див. рис. 5.61) визначає простий пріоритетний режим з фіксованими пріоритетами, одиничне — з циклічними пріоритетами. Розряд *D*3 керує станом лінії *SP*/*EN*. Розряд *M* /$\overline{S}$ (Master/Slave) (*D*2) дорівнює одиниці для ведучого ПКП і нулю для веденого. Розряд *D*1 задає автоматичне (0) або спеціальне закінчення переривання, а розряд *D*0 дорівнює одиниці в системі з МП *i*8086 і нулю в системі з МП *i*8080.

Після запису керуючих слів ініціалізації ПКП підготовлений до приймання запитів переривання і може працювати в *режимі з фіксованими пріоритетами запитів.* У цьому режимі запити впорядковані за пріоритетами: вхід *IRQ*0 має вищий пріоритет, а *IRQ*7 — нижчий. Цей режим ще назива-

Таблиця 5.17. **Молодший байт адреси команди *CALL***

Вхід запиту	Адресний інтервал, 4 байти								Адресний інтервал, 8 байт							
	D7	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*	*D7*	*D6*	*D5*	*D4*	*D3*	*D2*	*D1*	*D0*
IRQ7	*A7*	*A6*	*A5*	1	1	1	0	0	*A7*	*A6*	1	1	1	0	0	0
IRQ6	*A7*	*A6*	*A5*	1	1	0	0	0	*A7*	*A6*	1	1	0	0	0	0
IRQ5	*A7*	*A6*	*A5*	1	0	1	0	0	*A7*	*A6*	1	0	1	0	0	0
IRQ4	*A7*	*A6*	*A5*	1	0	0	0	0	*A7*	*A6*	1	0	0	0	0	0
IRQ3	*A7*	*A6*	*A5*	0	1	1	0	0	*A7*	*A6*	0	1	1	0	0	0
IRQ2	*A7*	*A6*	*A5*	0	1	0	0	0	*A7*	*A6*	0	1	0	0	0	0
IRQ1	*A7*	*A6*	*A5*	0	0	1	0	0	*A7*	*A6*	0	0	1	0	0	0
IRQ0	*A7*	*A6*	*A5*	0	0	0	0	0	*A7*	*A6*	0	0	0	0	0	0

ють *режимом повного вложення підпрограм оброблення переривань*. Якщо запити переривань надійдуть у ПКП одночасно, то буде обслуговуватися запит, який має більший пріоритет. Якщо під час обробки одного запита надійшов другий з більш високим пріоритетом, то МП переривае обробку поточного запиту і викликає підпрограму оброблення нового запиту. Після закінчення його оброблення МП продовжує перервану підпрограму оброблення. Запити з нижчими пріоритетами не переривають підпрограм оброблення запитів з вищими.

Для завдання інших режимів функціонування ПКП слід використовувати операційні слова *OCW*, що завантажуються після слів ініціалізації у будь-який поточний момент часу.

Операційне слово *OCW*1 (рис. 5.62, *а*) здійснює встановлення або скидання розрядів регістра *RGM*. Встановлення деякого розряду регістра маскування приводить до заборони переривання на відповідному входу. Операційне слово *OCW*2 (рис. 5.62, *б*) здійснює скидання розряду регістра *ISR* і циклічний зсув пріоритетів запитів. Можливі варіанти слова *OCW*2 і його функції наведено в табл. 5.18.

Якщо обслуговування запиту переривання необхідно закінчити скиданням розряду регістра *ISR* з вищим пріоритетом, то використовується слово *OCW*2 із значеннями *EOI* = 1 і *SEOI* = 1. За *EOI* = 1 і *SEOI* = 1 призначений для скидання, рівень переривання вказується у команді розрядами *D*2(*L*2)—*D*0(*L*0). Циклічний зсув пріоритетів задається у команді *OCW*2 розрядом *D*7(*R*).

У циклічному режимі використовується круговий порядок призначення пріоритетів. Останньому запиту, що обслуговується, присвоюється нижчий пріоритет, наступному — вищий. Пріоритети інших запитів циклічно зміщуються за

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	MT	M6	M5	M4	M3	M2	M1	M0

а

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	K	SEO	EOI	0	0	L2	L1	L0

б

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	X	ESMM	SMM	0	0	P	ERIS	RIS

Дозволу немає	x	0
Скидання режиму	1	0
Встановлення режиму	1	0

Заборона читання	x	0
Читання *RGI*	1	0
Читання *ISR*	1	0

в

Рис. 5.62. Формати операційних керуючих слів:
а – *OCW*1; *б* – *OCW*2; *в* – *OCW*3

шкалою пріоритетів. Якщо $R = 1$ і $SEOI = 0$, команда *OCW*2 присвоює нижчий пріоритет запиту з вищим пріоритетом, а якщо $R = 1$ і $SEOI = 1$, нижчий пріоритет присвоюється запиту, номер якого вказується розрядами $D2(L2) - D0(L0)$. Слово *OCW*2, як правило, записується у ПКП наприкінці підпрограми обслуговування переривання перед командою повернення з підпрограми *RET*.

Операційне слово *OCW*3 (рис. 5.62, *в*) дозволяє задати режим спеціального маскування (розряди *D*6, *D*5), режим опитування (розряд *D*2) і виконати зчитування стану ПКП (розряди *D*1, *D*0).

Режим спеціального маскування дає змогу на деякій ділянці програми вибірково керувати запитами з різними пріорите-

Таблиця 5.18. **Варіанти слова *OCW2***

Розряд команди								Режим пріоритетів
D7 (*R*)	*D6* (*SEOI*)	*D5* (*E0I*)	*D4*	*D3*	*D2*	*D1*	*D0*	
0	0	1	0	0	*x*	*x*	*x*	Фіксований пріоритет: *IRQ0* — вищий, *IRQ7* — нижчий
1	0	1	0	0	*x*	*x*	*x*	Циклічний зсув пріоритетів — запиту, що обслуговується, присвоюється нижчий пріоритет
0	1	1	0	0	*L2*	*L1*	*L0*	Фіксований пріоритет; *L2*—*L0* — номер розряду, що скидається
1	1	1	0	0	*L2*	*L1*	*L0*	Циклічний зсув пріоритетів; *L2*—*L0* — номер розряду, скидання якого відбувається у регістрі *ISR* (присвоєння йому нижчого пріоритету)
1	1	0	0	0	*L2*	*L1*	*L0*	Циклічний зсув пріоритетів без завершення переривання; *L2*—*L0* — номер входу *IRQ* з нижчим пріоритетом

тами і дозволяти переривання виконуваної програми навіть від входів з меншими пріоритетами. Режим спеціального маскування задається словом *OCW3* за *ESMM* = 1 і *SMM* = 1 та відміняє цей режим словом з *ESMM* = 1 і *SMM* = 0.

Режим опитування встановлюється за словом *OCW3*, якщо *P* = 1. У цьому режимі ПКП приймає запити і формує слово стану опитування, що містить номер запиту з найвищим пріоритетом (розряди *D2*—*D0* слова стану). Запит обслуговується на вимогу програми, яка здійснює за допомогою команди введення *IN* (*A* = 0, *RD* = 0) читання слова стану, програмне декодування його і перехід до відповідної підпрограми обслуговування переривання.

Вміст регістрів *RGI* і *ISR* зчитується після завантаження у ПКП слова *OCW3* з відповідними значеннями *ERIS* і *RIS* (див. рис. 5.62, *в*) з наступним виконанням команди введення *IN* (*A0* = 1 і $\overline{RD}$ = 0). Вміст регістра *RGM* зчитується без попереднього завантаження за командою *IN* (*A0* = 1, $\overline{RD}$ = 0).

Приклад 5.17. Запрограмувати контролери переривань з каскадним включенням (рис. 5.63) у системі з МП 80*x*86 (в PC IBM AT) на режим з фіксованими пріоритетами, не автоматичним закінченням переривань. Запрограмувати номер переривань для *IRQ*0 ведучого контролера — 08, для веденого — 70*H*. Адреси ведучого контролера 20*H* та 21*H*, веденого 0*A*0*H*, 0*A*1*H*. Визначити номери всіх переривань та їхні пріоритети.

Згідно з рис. 5.58, керуюче слово ініціалізації *ICW*1 дорівнює 11*H* для обох ПКП. Запис керуючого слова відбувається, якщо *A*0 = 0, тобто для адреси 20*H* для першого ПКП і 70*H* для другого.

Керуюче слово ініціалізації *ICW*2, згідно з рис. 5.59, має вигляд

0 0 0 0 1 *x x x* для ПКП1 і 01110 *x x x*, тобто відповідно 08 та 70*H*.

Керуюче слово ініціалізації *ICW3*, необхідне під час каскадного з'єднання, згідно з рис. 5.60, *а*, для ведучого контролера дорівнює 04. Єдина одиниця в цьому слові визначається підключенням веденого контролера до входу *IRQ*2. Керуюче слово ініціалізації *ICW*3 для веденого контролера визначається номером 2, до якого підключений вихід *INT* ПКП 2 (див. рис. 5. 63).

Керуюче слово ініціалізації *ICW4*, згідно з рис. 5.61, дорівнює 1*DH* для ПКП 1 і 09*H* для ПКП2.

Програма ініціалізації ПКП1 і ПКП2 така:

```
MOV AL,11H      ; Формування в AL керуючого слова ініціалізації
                ; ICW1
OUT 20H,AL      ; Пересилання його в ПКП1
OUT 0A0H,AL     ; і в ПКП2
MOV AL,08H      ; Формування в AL керуючого слова ініціалізації
                ; ICW2
OUT 21H,AL      ; Пересилання його в ПКП1
MOV AL,70H      ; Формування в AL керуючого слова ініціалізації
                ; ICW2
OUT 0A1H,AL     ; Пересилання його в ПКП2
MOV AL,04H      ; Формування в AL керуючого слова ініціалізації
                ; ICW3
```

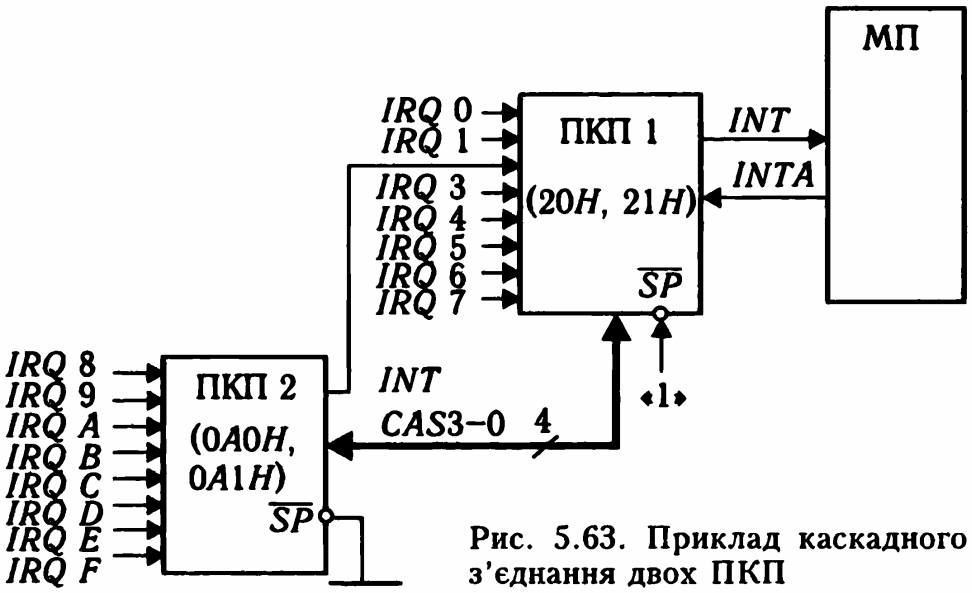


Рис. 5.63. Приклад каскадного з'єднання двох ПКП

```
OUT 21H,AL      ; Пересилання його в ПКП1
MOV AL,02H      ; Формування в AL керуючого слова ініціалізації
                ; ICW2
OUT 0A1H,AL     ; Пересилання його в ПКП2
MOV AL,0DH      ; Формування в AL керуючого слова ініціалізації
                ; ICW4
OUT 21H,AL      ; Пересилання його в ПКП1
MOV AL,09H      ; Формування в AL керуючого слова ініціалізації
                ; ICW4
OUT 0A1H,AL     ; Пересилання його в ПКП2
```

Номери переривань та їхню пріоритетну схему подано в табл. 5.19.

Зазначимо, що в схемі з двома ПКП (див. рис. 5.63) максимальна кількість запитів дорівнює 15. Це пов'язано з тим, що один із входів ведучого контролера з'єднаний з виходом веденого.

Приклад 5.18. Заборонити оброблення переривання *IRQ*2. Прийняти адресу ПКП 1, як у прикладі 5.18.

Для того щоб заборонити переривання, слід записати одиницю у відповідний розряд регістра маски як на рис. 5.6, *а*.

Програма має такий вигляд:

```
IN AL,21H       ; Прочитати вміст регістра маски
OR AL,04        ; Встановити розряд D2 у стан 1
MOV AL,70H      ; Переслати в ПКП1
```

Приклад 5.19. Дозволити оброблення переривання *IRQ*2. Прийняти адресу ПКП 1, як у прикладі 5.18.

Таблиця 5.19. **Пріоритети переривань**

Схема пріоритетів	Номер входу переривання	Команда переривання
Найвищий ↑	*IRQ* 0	*INT* 8
	IRQ 1	*INT* 9
	IRQ 2	*INT* 0*AH*
	IRQ 8	*INT* 70*H*
	IRQ 9	*INT* 71*H*
	IRQ 0*AH*	*INT* 72*H*
	IRQ 0*BH*	*INT* 73*H*
	IRQ 0*CH*	*INT* 74*H*
	IRQ 0*DH*	*INT* 75*H*
	IRQ 0*EH*	*INT* 76*H*
	IRQ 0*FH*	*INT* 77*H*
	IRQ 3	*INT* 0*BH*
	IRQ 4	*INT* 0*CH*
	IRQ 5	*INT* 0*DH*
	IRQ 6	*INT* 0*EH*
Найнижчий	*IRQ* 7	*INT* 0*FH*

Для того щоб дозволити переривання, треба записати нуль у відповідний розряд регістра маски як на рис. 5.6, *а*.

Програма має такий вигляд:

```
IN AL,21H          ; Прочитати вміст регістра маски
AND AL,11111011B   ; Встановити розряд D2 у стан 1
MOV AL,70H         ; Переслати в ПКП1
```

Приклад 5.20. Скласти підпрограму введення даних з порту з адресою 300*H* в комірку пам'яті з адресою *DS*:*SI* за перериванням.

Підпрограма оброблення переривання:

```
;Зберігання регістрів МП у стеку
PUSH AX            ; Переслати AX у стек
PUSH BX            ; Переслати BX у стек
...
MOV DX,300H        ; Записати в DX адресу порту
IN AL, DX          ; Ввести дані
MOV [SI],AL        ; Переслати у комірку пам'яті
...
; Закінчення переривання
MOV AL, 20H        ; запис в AL керуючого слова OCW2
OUT 20H,AL
; Повернення вмісту регістрів МП зі стеку (у зворотному порядку)
POP BX
POP AX
STI                ; Дозвіл переривань
IRET               ; Повернення з підпрограми оброблення перери-
                   ; вання
```

5.8. Приклад розробки мікропроцесорної системи

Нехай треба розробити принципову схему і програмне забезпечення МПС керування широтно-імпульсного стабілізатора напруги з частотою f = 1 кГц. Мікропроцесорна система керування забезпечує оброблення сигналу зворотного зв'язку U із виходу АЦП за законом пропорційного регулятора:

$$\gamma = 0,5 + (U - 20H)/256,$$

де γ — коефіцієнт заповнення імпульсів широтно-імпульсного стабілізатора; 20*H* — код опорного сигналу.

Структурну схему МПС керування зображено на рис. 5.64.

Сигнал зворотного зв'язку надходить на АЦП, де він перетворюється на двійковий код. Цей код по системній шині надходить у ЦП,

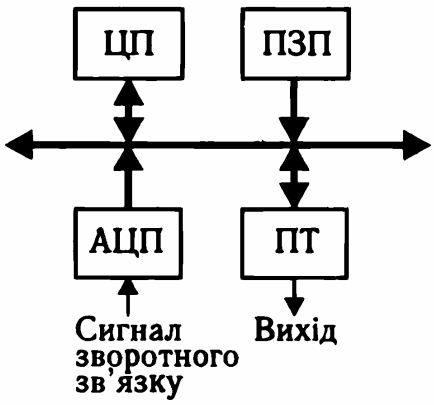


Рис. 5.64. Структурна схема МПС керування

де за програмою, записаною у ПЗП, обробляється за законом пропорційного регулятора. Результатом обчислення є коефіцієнт заповнення, який завантажується у регістр таймера як константа. На виході таймера отримуємо логічні рівні імпульсів керування ШІП.

Функціональну схему мікропроцесорної системи керування показано на рис. 5.65.

Під час розробки функціональної схеми центрального процесора виникає потреба у демультиплексуванні шини адреси-даних, буферизації шин адреси (AB) і шин даних (BD), а

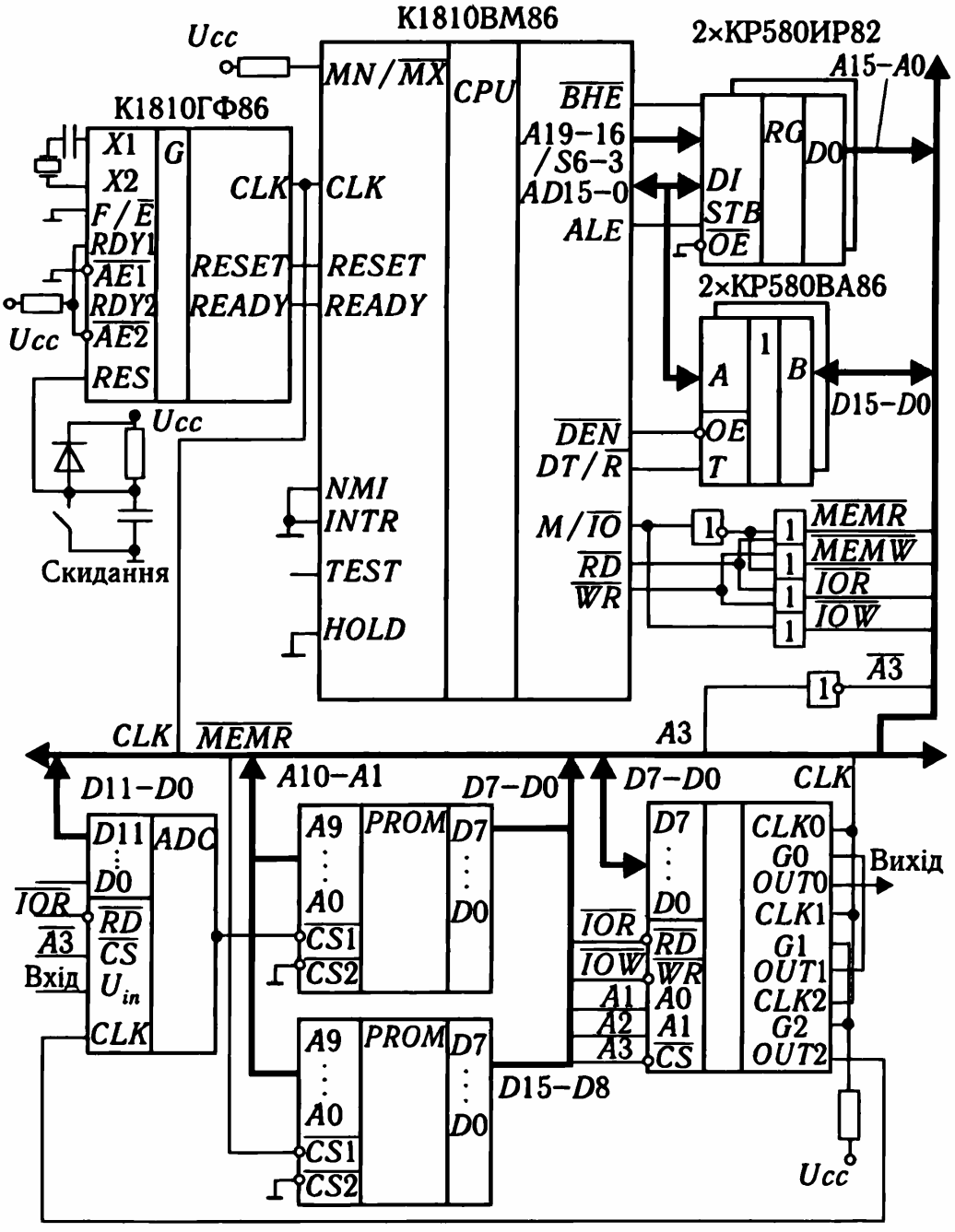


Рис. 5.65. Функціональна схема МПС

також у формуванні системних керуючих сигналів пам'яті та зовнішніх пристроїв. Демультиплексування здійснюється за допомогою двох ВІС К1810ИР82, що виконують функції фіксатора адреси і буфера шини *AB*, буферизація шин даних — за допомогою двоспрямованих шинних формувачів К1810ВА86, які підсилюють сигнали шини даних і формування керуючих сигналів — за допомогою комбінаційних логічних елементів. На виході цих елементів формуються сигнали $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$, $\overline{IOW}$.

Оскільки у розроблюваній МПС не потрібні режими ПДП, переривань та обміну за сигналом готовності, то порівняно зі схемою модуля ЦП (див. рис. 2.40) схема на рис. 5.65 не містить сигналів переривань, готовності, запиту ПДП і дозволу шин *BUSEN*. Тому на входи ВІС генератора, ЦП, регістрів-фіксаторів та буферних регістрів подаються постійні логічні рівні нуля або одиниці. Відсутні також кінцеві каскади схеми формування керуючих сигналів (у цій схемі немає потреби переводити керуючий сигнал у третій стан).

Модуль ПЗП виконано на базі двох ВІС КР556РТ5 ємністю 512 × 8 біт кожна. Пам'ять організована у вигляді двох банків пам'яті — молодшого і старшого. Молодший банк вмикається до молодшої половини шини даних $D7-D0$ і містить лише комірки пам'яті з парними адресами; старший банк — до старшої половини шини даних $D15-D8$ і містить лише комірки пам'яті з непарними адресами. Зчитування з ПЗП відбувається під час виконання циклу читання пам'яті. При цьому формується сигнал $\overline{MEMR} = 0$, який і переводить виходи ВІС ПЗП в активний стан. Із ПЗП завжди зчитується слово. Для цього прикладу початкову адресу ПЗП визначимо за нульових значень $A9-A1$, $A0$, а кінцеву — за одиничних.

Отже, початкова адреса ПЗП — $00000H$, кінцева адреса ПЗП — $003FFH$.

Функціональна схема містить також АЦП К572ПВ1, який є 12-розрядним перетворювачем напруги на двійковий код низької швидкодії. Оскільки АЦП має внутрішній регістр із входом керування третім станом, зовнішній порт введення не потрібний. Вихід АЦП з'єднаний з лініями $D11-D0$. З погляду процесора АЦП є 16-розрядним портом. Адреса 16-розрядного порту має бути парною. Як видно з рис. 5.65, АЦП вибирається за $A3 = 1$. Тому адреса АЦП буває будь-якою за $A3 = 1$, $A0 = 0$. Наприклад, оберемо адресу АЦП, що дорівнює $08H$.

Програмований таймер К1810ВИ54 у схемі (див. рис. 5.65) призначений для генерації імпульсів керування широтно-імпульсним стабілізатором. Таймер містить три незалежних

канали, кожний з яких може бути запрограмований на роботу в одному з шести режимів для двійкового та двійково-десяткового обчислення. У цьому прикладі використовують такі режими роботи каналів:

- канал 0 — програмований мультивібратор;
- канал 1 — імпульсний генератор частоти для запуску каналу 0;
- канал 2 — імпульсний генератор для задавання частоти роботи АЦП.

Як видно з рис. 5.65, таймер вибирається за адреси з $A3 = 0$. Лінії $A1$ і $A2$ обирають один з трьох каналів таймера або регістр керуючого слова. Отже, адресами таймера є:

- адреса каналу 0 — $00H$;
- адреса каналу 1 — $02H$;
- адреса каналу 2 — $04H$;
- адреса $RWCH$ — $06H$.

Розрахуємо константи завантаження таймера так. Перетворимо коефіцієнт заповнення широтно-імпульсного стабілізатора *gamma* на константу перерахунку, що завантажується у таймер. Зауважимо, що вихідна частота стабілізатора дорівнює 1 кГц, а частота тактових імпульсів $f_{CLK} = 5$ МГц.

Канал 0. Згідно із завданням коефіцієнт заповнення імпульсів широтно-імпульсного стабілізатора

$$gamma = 0{,}5 + \frac{1}{256}(U - 20H).$$

Визначимо період роботи широтно-імпульсного стабілізатора, мс:

$$T = \frac{1}{f} = \frac{1}{10^3} = 1.$$

Тоді тривалість імпульсу, мс:

$$\tau_i = \frac{T}{2} + \frac{T}{256}(U - 20H) = 0{,}5 + \frac{1}{256}(U - 20H).$$

Тривалість лічильних імпульсів каналів (CLK) за частоти роботи процесора 5 МГц, нс:

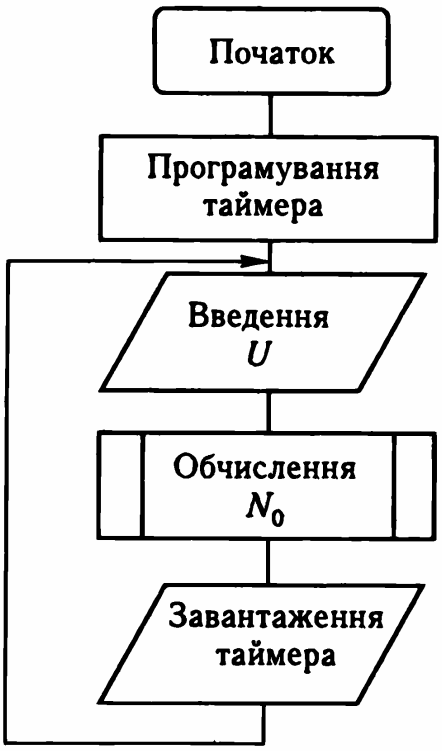
$$T_{CLK} = 1/(5 \cdot 10^6) = 200.$$

Код завантаження каналу 0 таймера визначають як відношення τ_i до T_{CLK}:

$$N_0 = \frac{0{,}5 \cdot 10^{-3}}{200 \cdot 10^{-9}} + \frac{10^{-3}}{256 \cdot 200 \cdot 10^{-9}}(U - 20H) \approx$$

$$\approx 2500 + 20(U - 20H).$$

Рис. 5.66. Алгоритм керування широтно-імпульсним стабілізатором



Канал 1. Коефіцієнт ділення каналу 1 визначають як відношення тривалості періоду широтно-імпульсного стабілізатора до тривалості періоду T_{CLK}:

$$N_1 = \frac{T}{T_{CLK}} = \frac{1000000}{200} = 5000.$$

Канал 2. Коефіцієнт ділення каналу 2 визначають як відношення тривалості періоду тактових імпульсів АЦП (170 кГц) до тривалості періоду T_{CLK}:

$$N_2 = \frac{T_{\text{АЦП}}}{T_{CLK}} = 29.$$

Алгоритм керування широтно-імпульсним стабілізатором показано на рис. 5.66. Спочатку в обидва канали таймера завантажуються керуючі слова. Керуюче слово для каналу 0 дорівнює 00110010*B* і визначає режим 1 лічильника 0, завантаження відбувається спочатку молодшого, а потім старшого байта, лічення — двійкове. Керуюче слово для першого і другого каналів визначає режим 1, завантаження спочатку молодшого, а потім старшого байтів, лічення — двійкове. Керуючі слова дорівнюють відповідно 01110100*B* і 10110100*B*.

Після завантаження керуючих слів програма завантажує початкові значення у канали 1 та 2 таймера. На цьому ініціалізація закінчується і починається нескінченний цикл введення коду зворотної напруги з АЦП, перерахування її на код завантаження і виведення його у канал 0.

Зазначимо, що після системного скидання регістри *CS* і *IP* МП відповідно набудуть значень 0*FFFFH* і 0000*H*. Це призведе до того, що в адресі першої комірки ПЗП, до якої після скидання звернеться МП у першому машинному циклі ВИБІРКА КОМАНДИ, чотири молодших розряди будуть нульовими, а всі інші — одиничними. Тоді ця адреса дорівнюватиме 3*F0H*.

Програма має такий вигляд:

```
        ORG 3F0H
        JMP START
        ORG 100H
START:  MOV AL, 00110010B   ; Формування в AL керуючого слова
                            ; режиму 1 каналу 0
```

```
    OUT 06, AL                  ; Виведення керуючого слова в
                                ; РКС
    MOV AL, 01110100B           ; Формування в AL керуючого
                                ; слова в РКС
    OUT 06, AL                  ; Виведення керуючого слова
                                ; режиму 2
                                ; каналу 1
    MOV AL,10110100B            ; Формування в AL керуючого
                                ; слова
                                ; режиму каналу 2
    OUT 06, AL                  ; Виведення керуючого слова
                                ; режиму РКС
    MOV AX, 5000                ; Задавання частоти перетворен-
                                ; ня f = 1 кГц
    OUT 02H, AL                 ; Запис молодшого байта коду
                                ; Попереднє встановлення в ка-
                                ; нал 1
    MOV AL, AH                  ; Запис старшого байта коду
                                ; Попереднє встановлення в ка-
                                ; нал 1
    OUT 02H,AL                  ; Задавання частоти АЦП f =
                                ; = 170 кГц
    MOV AL,29                   ; Запис коду, попереднє встанов-
                                ; лення в канал 2
    OUT 04H, AL
    MOV AL, 00
    OUT 04H, AL
L:  IN AX,08H                   ; Введення сигналу зворотного
                                ; зв'язку U;
    AND AX,0000 1111 1111 1111B ; Виділення значущих цифр U
    SUB AX,20H                  ; Віднімання U — 20H, резуль-
                                ; тат — у AX;
    MOV BL,20
    MUL BL                      ; Множення на 20:20(U—20H),
                                ; результат— в AX
    ADD AX,2500                 ; Додавання 2500+20(U—20H),
                                ; результат — в AX
    OUT 00H,AL                  ; Запис молодшого байта коду
                                ; Попереднє встановлення в ка-
                                ; нал 0
    MOV AL,AH
    OUT 00H,AL                  ; Запис старшого байта коду
                                ; Попереднє встановлення в ка-
                                ; нал 0
    MOV CX,100                  ; Затримка на час, більший ніж
                                ; тривалість
                                ; перетворення АЦП
D:  LOOP D                      ; t = 16 · 100 · 200 (нс)
    JMP L                       ; Перейти за адресою з міткою
                                ; L (цикл);
    END.
```

Контрольні запитання

1. Як забезпечується зберігання інформації, що надходить від ПВВ?
2. Назвіть способи адресування портів введення-виведення.
3. Дайте порівняльну характеристику видів обміну.
4. Які є типи програмного обміну?
5. Наведіть структурну схему обміну за стробом готовності.
6. Наведіть структурну схему обміну за перериванням.
7. Наведіть структурну схему обміну в режимі ПДП.
8. Як відбувається запам'ятовування вмісту акумулятора, РЗП, програмного лічильника та прапорців під час обміну за перериванням?
9. В яких випадках доцільно застосовувати прямий доступ до пам'яті?
10. Яке призначення ВІС програмованого паралельного інтерфейсу КР580ВВ55?
11. Опишіть режими роботи програмованого паралельного інтерфейсу.
12. Назвіть можливі комбінації ввімкнення портів ВІС КР580ВВ55.
13. Які порти паралельного інтерфейсу можуть працювати в усіх можливих режимах?
14. Запишіть керуюче слово для роботи паралельного інтерфейсу в режимі 0 під час налагодження портів *A* і *B* на виведення, а порту *C* — на введення.
15. Який принцип скидання-встановлення розрядів порту *C*?
16. Яке призначення ВІС програмованого інтерфейсу клавіатури та індикації?
17. Яку максимальну кількість клавіш можна з'єднати з ВІС програмованого інтерфейсу клавіатури та індикації КР580ВВ79?
18. Поясніть особливості режиму опитування матриці клавіатури.
19. Поясніть особливості режиму опитування матриці датчиків.
20. Поясніть особливості режиму введення за стробом.
21. Які функції виконує програмований таймер у МПС?
22. З яких блоків складається таймер? Що входить до складу одного лічильника?
23. Опишіть режими роботи таймера.
24. Чим відрізняється дія сигналу *GATE* в режимах 0 та 1?
25. Які нові функції має ВІС таймера КР1810ВИ54 порівняно з ВІС таймера КР580ВИ53?
26. Назвіть можливі комбінації роботи каналів таймера.
27. Назвіть приклади використання таймера МПС.
28. Запишіть керуючі слова для роботи таймерів у режимах 0, 2, 5.
29. Як можна прочитати вміст внутрішніх регістрів таймера?
30. У чому полягає відмінність між режимами 4 і 5?
31. В якому режимі таймер працює як подільник частоти?
32. Яке максимальне число можна завантажити у внутрішні регістри таймера?
33. Для чого потрібні лінії *A*0, *A*1 таймера?
34. Які функції виконує КПДП у мікропроцесорній системі?
35. Розкажіть про режими роботи КПДП.
36. В яких випадках доцільно застосовувати прямий доступ до пам'яті?
37. Яке призначення регістра станів?
38. Який принцип призначення пріоритетів каналів?

39. Які функції виконує порт послідовного передавання даних у *VGC*?
40. Розкажіть про режими роботи ВІС.
41. Поясніть роботу ВІС послідовного інтерфейсу в асинхронному режимі.
42. Поясніть роботу ВІС послідовного інтерфейсу в синхронному режимі.
43. Як здійснюється обмін даними за синхронного режиму з внутрішньою і зовнішньою синхронізацією?
44. Які функції виконує контролер переривань у МПС керування?
45. Яку кількість запитів переривань може обслужити одна ВІС ПКП?
46. Якої максимальної кількості запитів переривань можна досягти у системі переривань на базі ВІС КР580ВН59?
47. Для чого призначений режим спеціального маскування?
48. Які пріоритетні схеми оброблення запитів переривань реалізуються у ВІС КР580ВН59?
49. Як змінюються пріоритети у схемі циклічного оброблення пріоритетів?
50. Коли доцільно використовувати режим фіксованих пріоритетів?
51. Коли доцільно використовувати режим циклічних пріоритетів?
52. Який із входів *IRQ* має вищий пріоритет?
53. Як можна зменшити пріоритет входів *IRQ*?
54. Як можна вимкнути одну або кілька ліній *IRQ*?
55. Коли можна записувати команди ініціалізації і керування?
56. Для чого необхідні сигнали *INT* і *INTA*?

Розділ 6

ОДНОКРИСТАЛЬНІ МІКРОКОНТРОЛЕРИ З CISC-АРХІТЕКТУРОЮ

Однокристальний мікроконтролер — пристрій, виконаний конструктивно в одному корпусі ВІС, що містить усі компоненти МПС: процесор, пам'ять даних, пам'ять програм, програмовані інтерфейси. Однокристальним мікроконтролерам (ОМК) притаманні такі особливості:

- система команд зорієнтована на виконання задач керування і регулювання;
- алгоритми, що реалізуються на ОМК, мають багато розгалужень залежно від зовнішніх сигналів;
- дані, з якими оперують ОМК, не повинні мати велику розрядність;
- схемна реалізація систем керування на базі ОМК нескладна і має невисоку вартість;
- універсальність і можливість розширення функцій керування значно нижчі, ніж у системах з однокристальними МП.

Однокристальні мікроконтролери є зручним інструментом для створення сучасних вбудованих пристроїв керування різним обладнанням, наприклад автомобільною електронікою, побутовою технікою, мобільними телефонами тощо.

6.1. Архітектура і функціональні можливості однокристальних мікроконтролерів

Структуру ОМК та функціонування основних блоків розглянемо на прикладі ВІС К1816ВЕ51 (*i*80*x*51) (рис. 6.1). Графічне позначення мікросхеми наведено на рис. 6.2.

Структурна схема ОМК містить: блок 8-розрядного центрального процесора ЦП; пам'ять програм ПЗП ємністю 4 Кбайт; пам'ять даних ОЗП ємністю 128 байт; чотири 8-розрядних програмованих порти введення-виведення *P*0—*P*3; послідов-

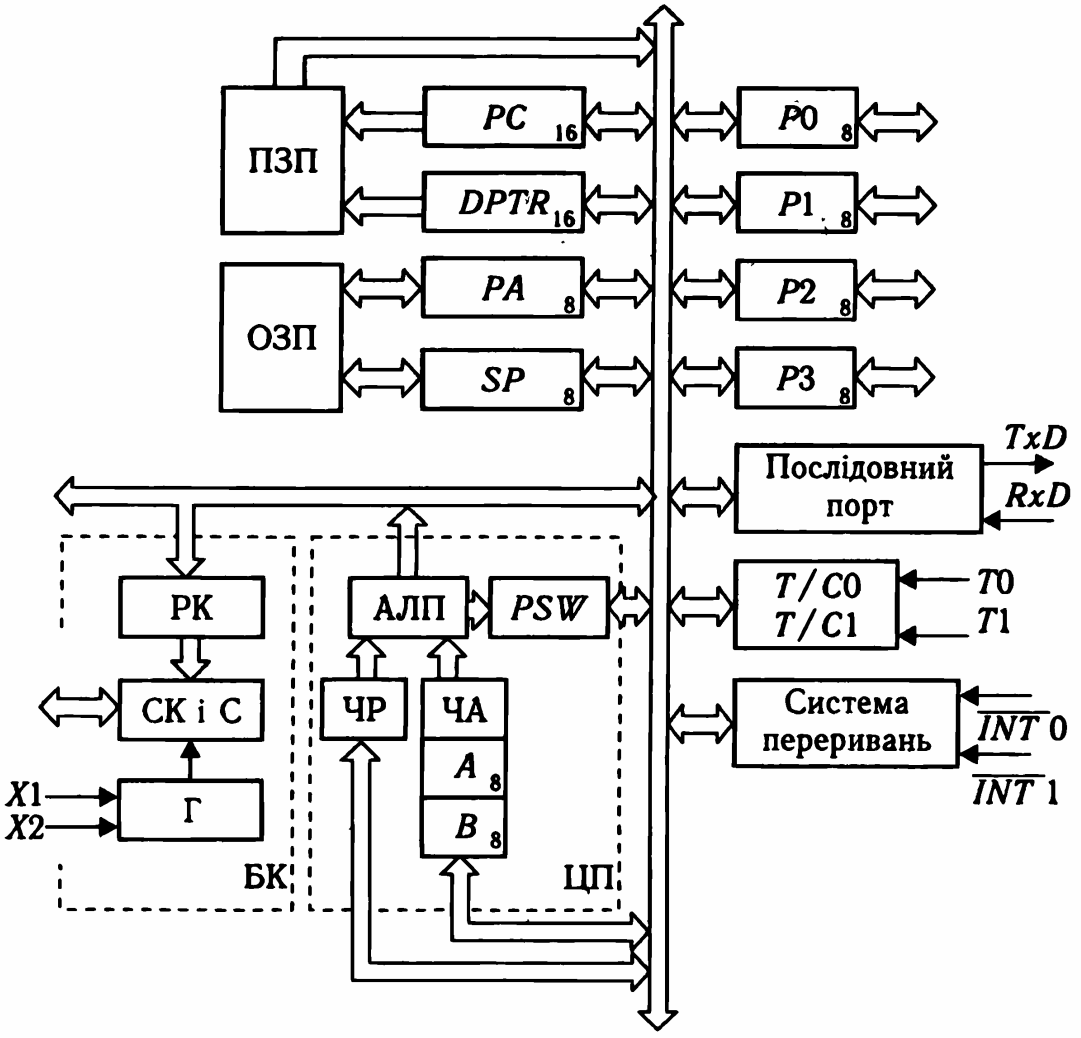
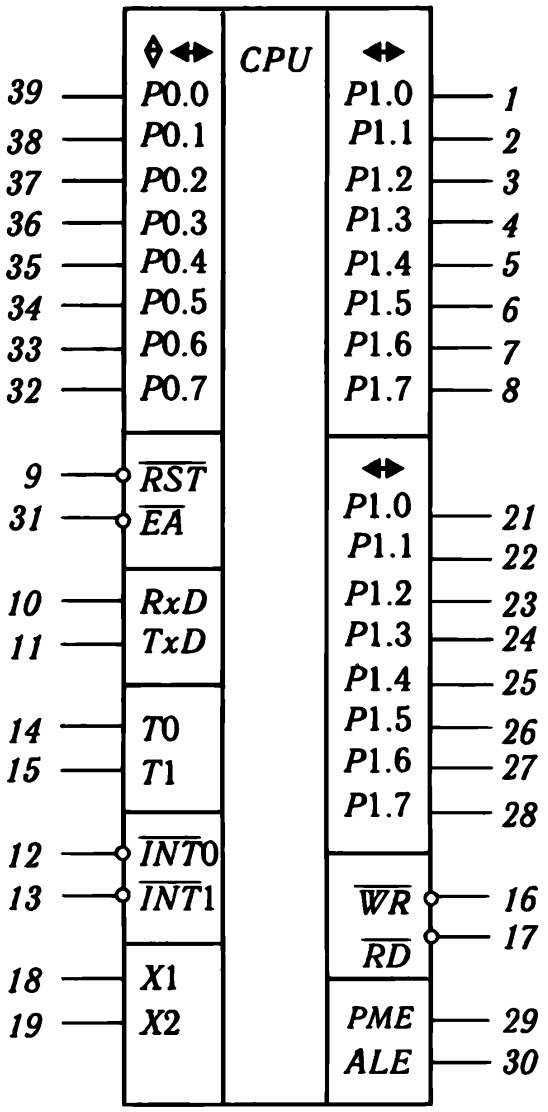


Рис. 6.1. Структурна схема ОМК К1816ВЕ51

ний порт; два 16-розрядних програмованих таймери/лічильники $T/C0$, $T/C1$; систему переривань з п'ятьма векторами і двома рівнями пріоритетів; блок керування (БК).

Блок ЦП містить 8-розрядний АЛП, два акумулятори A і B, регістр слова стану процесора PSW (*Processor State Word*) та програмно-недоступні буферні регістри ЧА і ЧР, що виконують функції розподілу вхідних та вихідних даних АЛП. Центральний процесор виконує операції додавання, віднімання, множення, ділення, логічні операції І, АБО, НЕ, ВИКЛЮЧАЛЬНЕ АБО, операції зсуву і скидання. Він оперує з такими типами змінних: булевими (1 біт), цифровими (4 біт), байтовими (8 біт) та адресними (16 біт). Характерною особливістю ОМК є великий набір операцій з бітами: окремі біти змінних можуть бути встановлені, скинуті, інвертовані, перевірені, передані. Це дає змогу легко реалізовувати алгоритми, що містять операції над булевими змінними типу «так-ні» (*«true-false»*).

Рис. 6.2. Графічне позначення ОМК К1816ВЕ51



КР1816ВЕ51 (20=*GND*, 40=+5В)

Акумулятор *A* є джерелом одного з операндів і місцем розміщення результату виконання багатьох команд. Ряд команд, наприклад, передавання інформації в/із ОЗП, команди *TEST*, *INC*, *DEC* можуть виконуватися без участі акумулятора. Акумулятор *B* використовується як акумулятор лише в командах множення і ділення, а в інших випадках — як РЗП.

Регістр слова стану процесора *PSW* зберігає інформацію про стан АЛП у процесі виконання програми і має формат, наведений у табл. 6.1.

Призначення прапорців *C*, *AC* аналогічне призначенню прапорців *CF*, *AF* у МП *i*8086. Прапорець *OV* встановлюється у командах додавання і віднімання, якщо результат перевищує ємність 7 біт і старший біт не може бути інтерпретований як знаковий; у командах ділення відбувається скидання *OV*, а під час ділення на нуль він знову встановлюється. У командах множення *OV* набуває значення логічної одиниці, якщо результат перевищує *0FFH*. Прапорець *P* є доповненням вмісту акумулятора *A* до парності, тобто 9-розрядне слово, що складається з 8 біт акумулятора *A* і біта *P*, має завжди парну кількість одиниць.

Постійний запам'ятовувальний пристрій, або **резидентна пам'ять програм (РПП)** має інформаційну ємність 4 Кбайт і виконаний у вигляді ПЗП програмною маскою. Інші ОМК, наприклад К1816751, мають ПЗП *EPROM* (див. табл. 6.1). ПЗП має 16-розрядну адресну шину, що дає змогу розширити пам'ять до 64 Кбайт через під'єднання зовнішніх ВІС ПЗП. Адреса визначається вмістом лічильника команд *PC* (*Program Counter*) або вмістом регістра-покажчика даних *DPTR* (*Data*

Таблиця 6.1. **Формат слова стану *PSW***

Біт	Позначення	Призначення	Доступ до біта
7	*C*	Прапорець перенесення	А або П
6	*AC*	Прапорець додаткового перенесення	А або П
5	*F0*	Прапорець користувача	П
4	*RS1*	Покажчик банку робочих регістрів:	
3	*RS0*	00 — банк 0; 10 — банк 2; 01 — банк 1; 11 — банк 3	П
2	*OV*	Прапорець переповнення	П
1	—	Резервний прапорець	П
0	*P*	Біт парності	А або П

Примітка. У таблиці 6.1 використано такі позначення: А — біт встановлюється апаратно; П — біт встановлюється програмно.

Pointer Register). Регістр *DPTR* використовується за непрямих переходів у програмі або під час адресації таблиць, або як один 16-розрядний регістр, або як два незалежних 8-розрядних регістри *DPH* і *DPL*.

Розподіл адресного простору ПЗП показано на рис. 6.3. Молодші адреси ПЗП відводяться для оброблення переривань і початку роботи ОМК після скидання.

Оперативний запам'ятовувальний пристрій, або **резидентна пам'ять даних (РПД),** (рис. 6.4) складається з двох областей. Перша область — ОЗП даних з інформаційною ємністю 128 × 8 біт з адресами $0-7FH$, друга область — регістри спеціальних функцій (*SFR — Special Function Registers*) з адресами $80H-FFH$. Перелік регістрів спеціальних функцій наведено в табл. 6.2.

Резидентна пам'ять даних адресується 8-розрядними регістром адреси (*PA*) або покажчиком стеку (*SP*) (див. рис. 6.1). Регістр адреси є програмно-недоступним регістром, у який завантажується адреса комірки ОЗП під час виконання команд. Регістр *SP* призначений для адресації стеку, який є частиною РПД. Вміст *SP* інкрементується перед запам'ятовуванням даних у стеку за командами *PUSH* і *CALL* та декрементується за командами *POP* і *RET*. Подібний спосіб адресації елементів стеку називають *передінкрементним-постдекрементним*. У процесі ініціалізації ОМК після надходження сигналу *RESET* у *SP* автоматично завантажується код $07H$. Отже, якщо програма не перевизначає стек, то перший байт даних у стеку розміщується у комірці РПД за адресою $08H$.

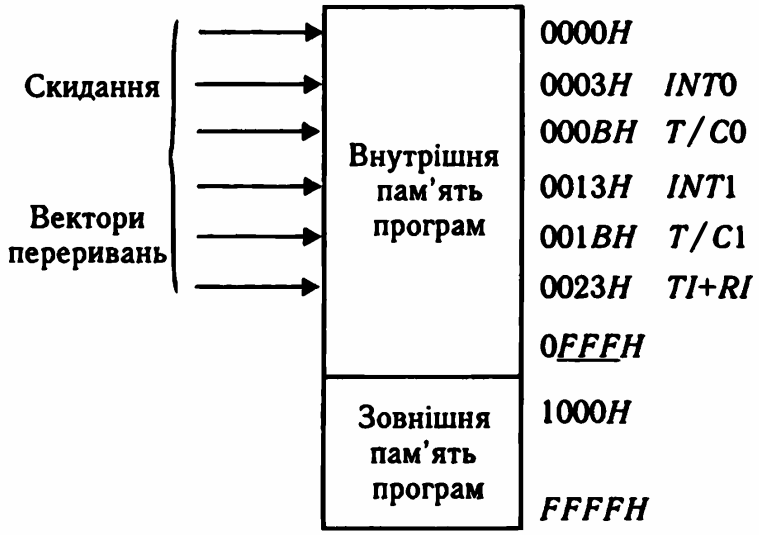


Рис. 6.3. Розподіл адресного простору РПП

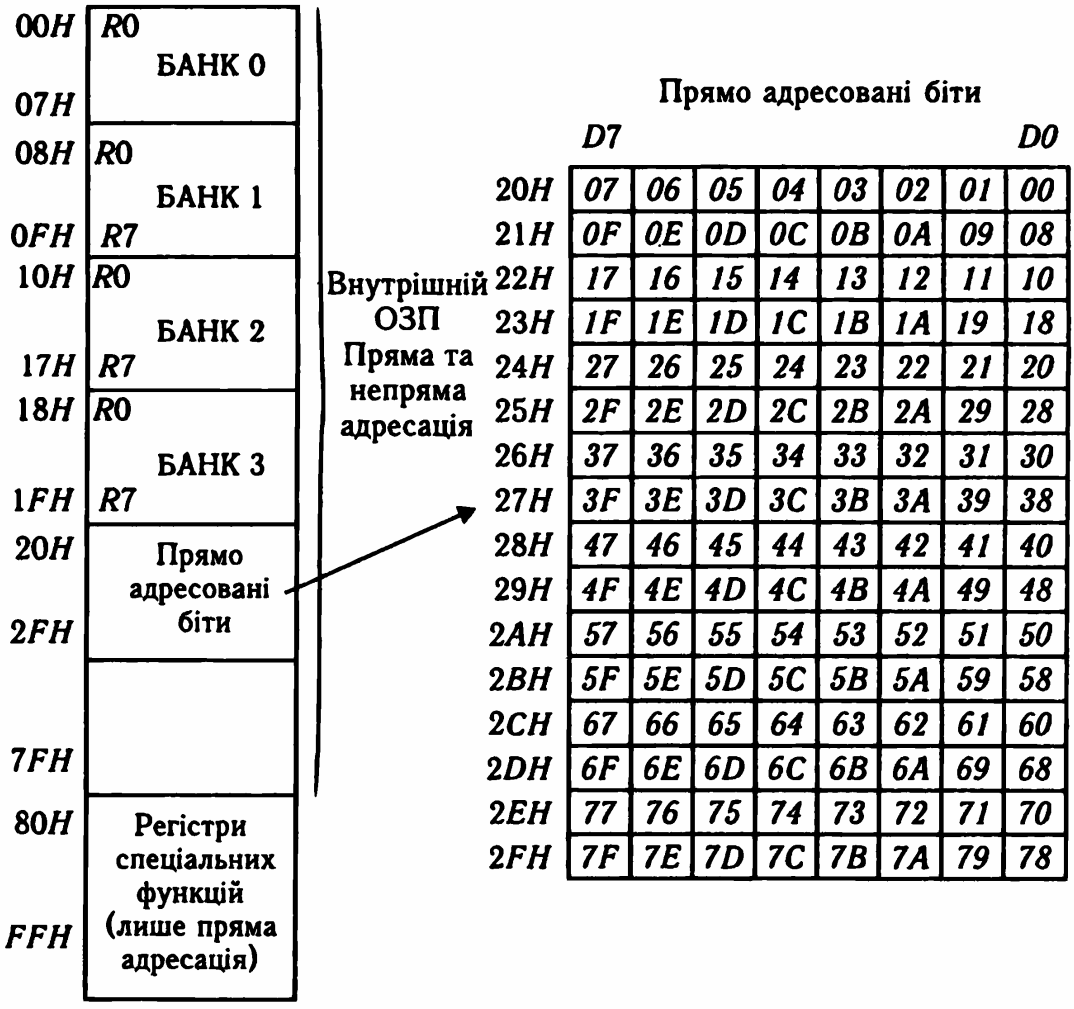


Рис. 6.4. Резидентна пам'ять даних

Таблиця 6.2. **Регістри спеціальних функцій**

Позначення	Найменування, адреси бітів	Адреса	Значення після скидання
*ACC**	Акумулятор *A* *E7 E6 E5 E4 E3 E2 E1 E0*	*0E0H*	00
*B**	Акумулятор *B* *F7 F6 F5 F4 F3 F2 F1 F0*	*0F0H*	00
*PSW**	Слово стану програми *D7 D6 D5 D4 D3 D2 D1 D0* *CY | AC | F0 | RS1 | RS0 | OV | | P*	*0D0H*	00
SP	Регістр-покажчик стеку	*81H*	07
DPTR	Регістр-покажчик даних: *DPH* – старший байт *DPL* – молодший байт	 *83H* *82H*	 00 00
*P0**	87 86 85 84 83 82 81 80	*80H*	*0FFH*
*P1**	Порт 1 97 96 95 94 93 92 91 90	*90H*	*0FFH*
*P2**	Порт 2 *A7 A6 A5 A4 A3 A2 A1 A0* *A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8*	*0A0H*	*0FFH*
*P3**	Порт 3 *B7 B6 B5 B4 B3 B2 B1 B0* $\overline{RD}$ | $\overline{WR}$ | *T1* | *T0* | $\overline{INT1}$ | $\overline{INT0}$ | *RxD* | *TxD*	*0B0H*	*0FFH*
*IP**	Регістр пріоритетів *BF BE BD BC BB BA B9 B8* *| | PT2 | PS | PT1 | PX1 | PT0 | PX0*	*0B8H*	*xx*000000*B*
*IE**	Регістр маски переривань *AF AE AD AC AB AA A9 A8* *EA | | ET2 | ES | ET1 | EX1 | ET0 | EX0*	*0A8H*	0*x*000000*B*
TMOD	Регістр режиму таймера-лічильника *G1 | C/$\overline{T1}$ | M1.1 | M0.1 | G0 | C/$\overline{T0}$ | M1.0 | M0.0*	*89H*	00

Продовж. табл. 6.2

Позначення	Найменування, адреси бітів	Адреса	Значення після скидання
*TCON**	Регістр керування-статусу таймерів *BF BE BD BC BB BA B9 B8* *TF1 TR1 TF0 TR0 IE1 IT1 IE0 IT0*	88*H*	00
TH0	Таймер 0 (старший байт)	8*CH*	00
TL0	Таймер 0 (молодший байт)	8*AH*	00
TH1	Таймер 1 (старший байт)	8*DH*	00
TL1	Таймер 1 (молодший байт)	8*BH*	00
*SCON**	Регістр керування приймачем-передавачем *9F 9E 9D 9C 9B 9A 99 98* *SM0 SM1 SM2 REN TB8 RB8 TI RI*	98*H*	00
SBUF	Буфер приймачів-передавачів	99*H*	*xx*
PCON	Регістр керування потужністю *SMOD GF1 GF0 PD IDL*	87*H*	0*xxxxxx*

*Примітка. Позначені регістри допускають адресацію окремих бітів.

Резидентна пам'ять данних, так само як і РПП, може бути розширена до 64 Кбайт під'єднанням зовнішніх ВІС.

Блок керування складається з генератора (Г) тактових сигналів, програмно-недоступного регістра команд (РК) та схеми керування і синхронізації (СК і С). Структурну схему блока керування показано на рис. 6.5. Код команди, зчитаної з РПП, запам'ятовується у 8-розрядному РК і надходить на дешифратор команд (ДШК), який входить до складу СК і С. Дешифратор команд формує 24-розрядний код, що надходить на програмовану логічну матрицю (ПЛМ), а після цього — на блок логіки керування.

Блок логіки керування на підставі декодованого коду команди, зовнішніх керуючих сигналів $\overline{RST}$ (сигналу загального скидання), $\overline{EA}$ (сигналу блокування роботи з РПП) та сигналів від внутрішнього формувача імпульсів синхронізації виробляє внутрішні сигнали керування.

Внутрішній формувач імпульсів синхронізації формує:

- внутрішні сигнали синхронізації машинних циклів;
- вихідний сигнал дозволу фіксації адреси *ALE*;
- сигнал дозволу програмної пам'яті *PME* (формується лише під час роботи із зовнішньою пам'яттю).

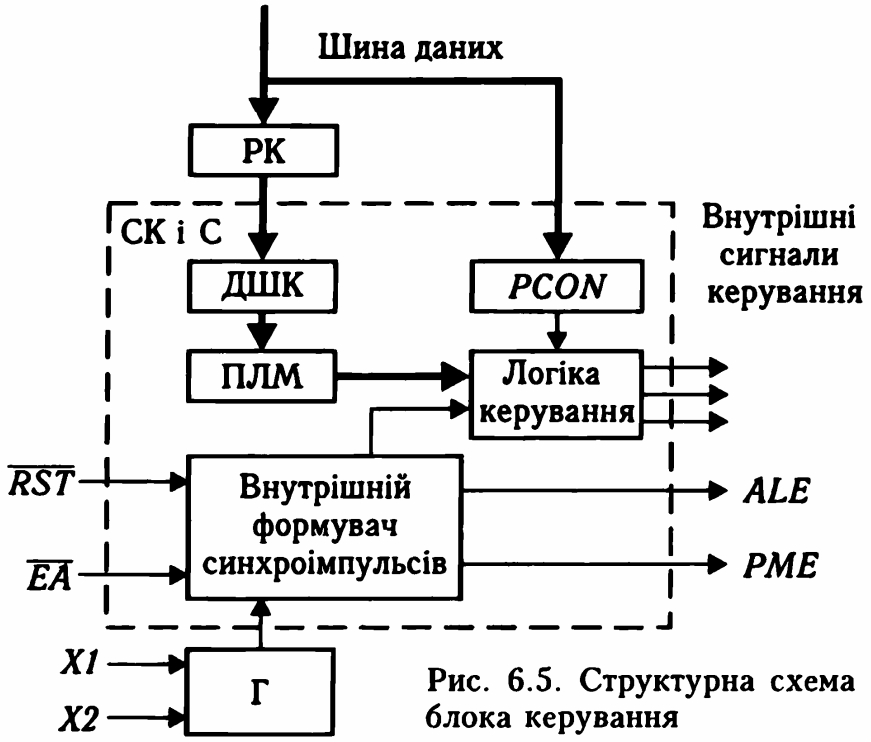


Рис. 6.5. Структурна схема блока керування

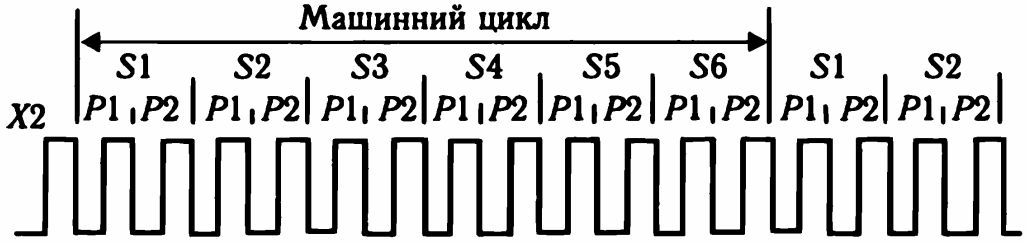


Рис. 6.6. Діаграма формування машинних циклів ОМК

Машинний цикл (рис. 6.6) має фіксовану тривалість і містить шість станів — $S1 - S6$, кожний з яких за тривалістю відповідає одному такту.

Кожний стан, або такт, складається з двох фаз — $P1$ і $P2$. Тривалість фази дорівнює періоду сигналу Q, який формується або вбудованим (внутрішнім) тактовим генератором (рис. 6.7) після під'єднання до виводів 18 ($X2$) та 19 ($X1$) ОМК кварцового резонатора (рис. 6.8, *а*) або LC-ланцюга (рис. 6.8, *б*), або зовнішнім джерелом тактових сигналів (рис. 6.9).

Частота імпульсів генератора для схеми на рис. 6.8, *а* визначається як

$$f = \frac{1}{2\pi\sqrt{LC'}},$$

де $C' = \frac{C + 3C_{вив}}{2}$ ($C_{вив} \approx 10$ пФ — ємність виводу).

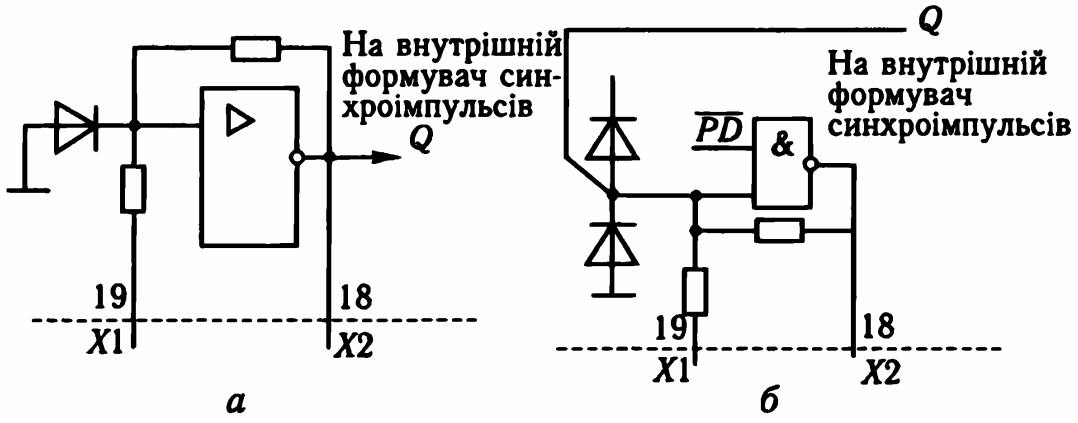


Рис. 6.7. Принципові схеми внутрішніх тактових генераторів:
а — *n*-МДН-технологія; *б* — КМДН-технологія

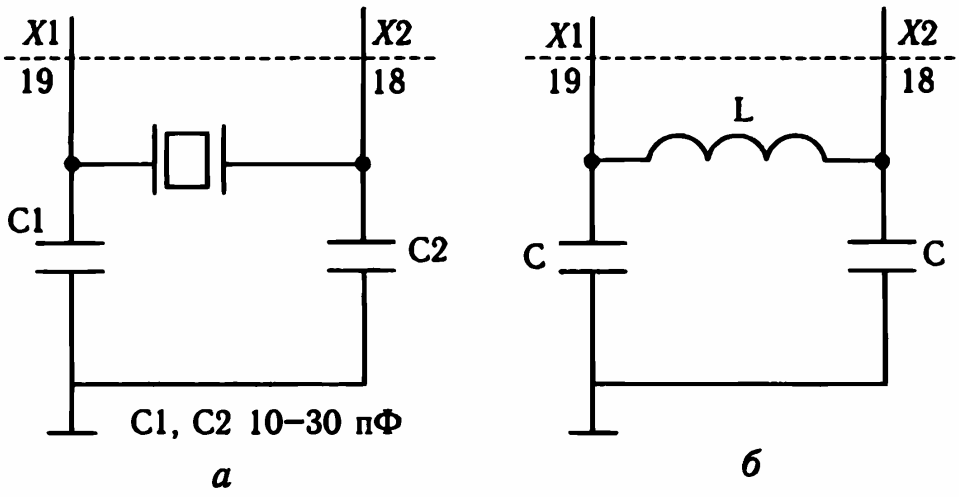


Рис. 6.8. Зовнішні ланцюги внутрішнього тактового генератора:
а — під'єднання кварцового резонатора; *б* — під'єднання LC-ланцюга

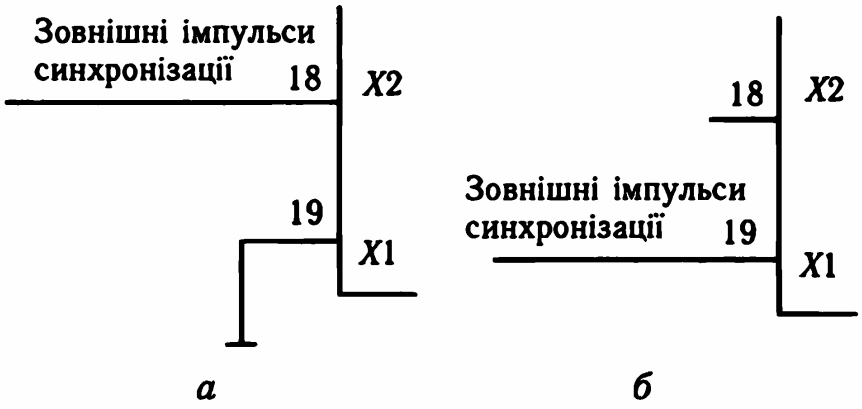


Рис. 6.9. Під'єднання зовнішнього джерела тактових сигналів:
а — для *n*-МДН; *б* — для КМДН

Під'єднання зовнішнього джерела тактових сигналів до ОМК, виконаних за *n*-МДН (див. рис. 6.9, *а*) та КМДН (див. рис. 6.9, *б*) технологіями, відрізняється тим, що у першому випадку зовнішні імпульси синхронізації надходять на виводи 18(*X*2) та 19(*X*1), а в другому — на виводи 19(*X*1) та 20(спільний); при цьому вивід 18(*X*2) залишається непід'єднаним.

За частоти кварцового резонатора або тактової частоти зовнішніх імпульсів синхронізації 12 МГц тривалість машинного циклу становить 1 мкс.

Параметри зовнішніх імпульсів синхронізації f_{max} = 12 мГц для частоти подано на рис. 6.10.

У блок керування входить також регістр керування споживанням *PCON* (*Power CONtrol*).

Порти введення-виведення *P0*—*P3* (див. рис. 6.1) призначені для забезпечення побайтного обміну інформацією ОМК із зовнішніми пристроями по 32 лініях введення-виведення. Принципові схеми ліній портів *P0*—*P3* відповідно показано на рис. 6.11, *а*—*г*. Кожна лінія порту містить керований регістр-фіксатор, два буфери і вихідний транзисторний каскад. Рівні вхідних і вихідних сигналів портів відповідають стандарту ТТЛ-логіки. Будь-яку лінію портів можна використовувати для введення або виведення інформації незалежно від інших ліній. Для того щоб лінія порту використовувалася для введення у відповідний *D*-тригер регістра-фіксатора має бути записана логічна одиниця, що закриває МДН-транзистор вихідного каскаду.

Фізичні адреси портів:

P0 — 80*H*, за бітової адресації 80*H*—87*H*;

P1 — 90*H*, за бітової адресації 90*H*—97*H*; *P2* — *A0H*, за бітової адресації *A0H*—*A7H*;

P3 — *B0H*, за бітової адресації *B0H*—*B7H*.

Порт *P0* є двонапрямленим, оскільки через нього можна у будь-який момент вводити та виводити інформацію. Виводи порту *P0* мають три стани. Через порт *P0*:

• виводиться молодший байт адреси *A7*—*A0* під час роботи із зовнішнім ПЗП і зовнішнім ОЗП;

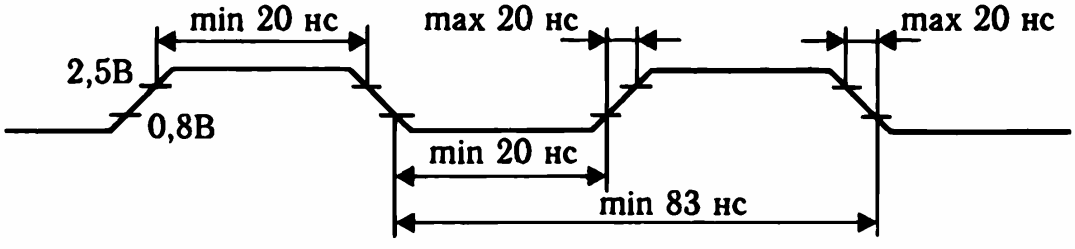


Рис. 6.10. Параметри зовнішніх імпульсів синхронізації

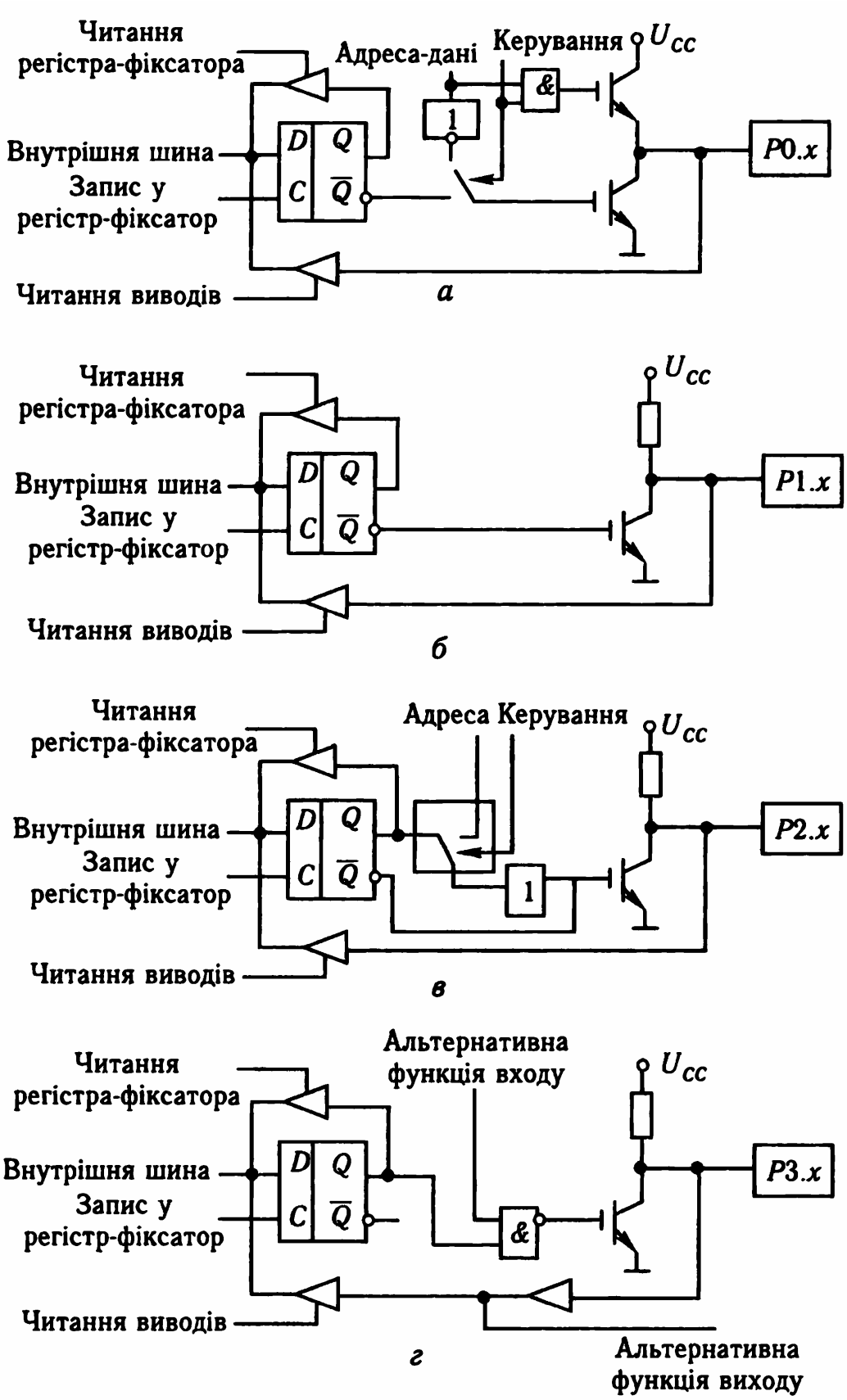


Рис. 6.11. Принципові схеми ліній портів:
а – *P*0; *б* – *P*1; *в* – *P*2; *г* – *P*3

• видається з ОМК та приймається в ОМК байт даних у процесі роботи із зовнішньою пам'яттю, при цьому обмін байтом даних та виведення молодшого байта адреси зовнішньої пам'яті мультиплексовано у часі;

• задаються дані для програмування внутрішнього ПЗП *EPROM*.

Порти *P*1—*P*3 є квазідвонапрямленими, оскільки у будь-який момент через порти можна лише виводити інформацію. Для введення інформації слід записати в усі розряди регістра-фіксатора логічні одиниці. Після цього можна виконувати введення.

Через порт *P*1: задається молодший байт адреси під час програмування внутрішнього ПЗП *EPROM* та під час читання внутрішнього ПЗП.

Через порт *P*2: виводиться старший байт адреси *A*15—*A*8 у процесі роботи із зовнішнім ПЗП і зовнішнім ОЗП тоді, коли адреса є 16-розрядною; задається старший байт *A*15—*A*8 адреси під час програмування внутрішнього ПЗП *EPROM* та читання внутрішнього ПЗП.

Порт *P*3 можна використовувати як для введення-виведення інформації, так і для реалізації альтернативних функцій обміну інформацією (див. рис. 6.11). Альтернативні функції наведені в табл. 6.3. Кожну з восьми ліній порту *P*3 користувач може запрограмувати на виконання альтернативних функцій записом логічної одиниці у відповідні біти регістра-замка (*P*3.0—*P*3.7) порту *P*3.

Послідовний порт (див. рис. 6.1) призначений для забезпечення послідовного обміну даними. Його можна використовувати або як регістр зсуву, або як *універсальний асинхронний приймач-передавач* з фіксованою або змінною швидкістю обміну та із можливістю дуплексного режиму. Послідовний порт може працювати в одному з чотирьох режимів: (режим 0, режим 1, режим 2, режим 3). Послідовний порт програмується на один з режимів через запис керуючого слова в регістр *SCON* (*Serial port CONtrol*). Призначення бітів регістра *SCON* наведено у табл. 6.4.

У режимі 0 послідовний порт — це 8-розрядний регістр зсуву. Байт інформації передається і приймається через вивід *RxD*, при цьому через вивід *TxD* видаються сигнали синхронізації зсуву. Приймання і видавання байта починається з молодшого розряду і закінчується старшим. Швидкість обміну фіксована і дорівнює $f_Q/12$, де f_Q — частота синхронізації ОМК.

На рис. 6.12, *а* показано функціональну схему послідовного порту в режимі «0», а на рис. 6.12, *б* — відповідні діаграми.

Таблиця 6.3. **Альтернативні функції порту *P3***

Біти регістра-замка порту *P3*		
Біт	Позиція	Альтернативна функція обміну інформацією
RxD	*P3.0*	Вхід приймача послідовного порту в режимах 1–3. Введення-виведення послідовних даних у режимі регістра зсуву
TxD	*P3.1*	Вихід передавача послідовного порту в режимах 1–3. Вихід синхронізації під час роботи послідовного порту у режимі 0
$\overline{INT0}$	*P3.2*	Вхід запиту переривання 0. Приймання сигналів низького рівня або зріз сигналу
$\overline{INT1}$	*P3.3*	Вхід запиту переривання 1. Приймається сигнал низького рівня або зріз сигналу
T0	*P3.4*	Вхід таймера-лічильника 0 або тестовий вхід 0
T1	*P3.5*	Вхід таймера-лічильника 1 або тестовий вхід 1
$\overline{WR}$	*P3.6*	Запис. Апаратне формування активного сигналу низького рівня у разі звернення до зовнішньої пам'яті даних
$\overline{RD}$	*P3.7*	Читання. Активний сигнал низького рівня формується апаратно у разі звернення до зовнішньої пам'яті даних

Таблиця 6.4. **Призначення бітів регістра *SCON***

Біти	Позначення	Призначення
7	*SM0*	Визначають один з чотирьох режимів роботи послідовного порту: *SM0 SM1* 0 0 – режим 0; 0 1 – режим 1; 1 0 – режим 2; 1 1 – режим 3.
6	*SM1*	
5	*SM2*	Дозвіл мультипроцесорної роботи: *у режимі 0 SM2* має бути скинутий у стан «0»; *у режимі 1* за *SM2* = 1 прапорець *RI* встановлюється у стан «1» після надходження стоп-біта, що дорівнює одиниці; *у режимах 2* і *3* за *SM2* = 1 прапорець *RI* дорівнює нулю, якщо дев'ятий біт прийнятих даних дорівнює нулю
5	*SM2*	Дозвіл мультипроцесорної роботи: *у режимі 0 SM2* має бути скинутий у стан «0»;

Продовж. табл. 6.4

Біти	Позначення	Призначення
		у режимі 1 за *SM2* = 1 прапорець *RI* встановлюється у стан «1» після надходження стоп-біта, що дорівнює одиниці; *у режимах 2 і 3* за *SM2* = 1 прапорець *RI* дорівнює нулю, якщо дев'ятий біт прийнятих даних дорівнює нулю
4	*REN*	Дозвіл прийняття послідовних даних, встановлення і скидання яких відбувається програмно відповідно для дозволу та заборони прийняття даних
3	*TB8*	Дев'ятий біт переданих даних у режимах 2 і 3. Встановлення і скидання відбувається програмно
2	*RB8*	*У режимі 0 RB8* не використовується; *у режимі 1* за *SM2* = 0 — прийнятий стоп-біт; *у режимах 2* і *3* — дев'ятий біт прийнятих даних
1	*TI*	Прапорець переривання передавача: *у режимі 0* — установлюється апаратно після видавання восьмого біта; *в інших режимах* — установлюється апаратно під час формування стоп-біта. Скидання відбувається програмно в усіх режимах
0	*RI*	Прапорець переривання приймача: *за SM2 = 0:* установлюється апаратно після прийняття восьмого біта в режимі 0 або через половину інтервалу стоп-біта в режимах 1, 2, 3; *за SM2 = 1:* в режимі 1 *RI* не встановлюється, якщо не прийнятий стоп-біт; дорівнює одиниці у режимах 2 та 3; *RI* не встановлюється, якщо дев'ятий прийнятий біт даних дорівнює одиниці. Скидання відбувається програмно в усіх режимах

Передавання починається з будь-якої команди, яка використовує буфер приймача-передавача *SBUF* як регістр призначення, наприклад *MOV SBUF, A.*

У фазі *P2* стану *S6* (*S6P2*) пристрій керування за сигналом «Запис у *SBUF*» записує байт у *регістр зсуву передавача*, встановлює *D-тригер* дев'ятого біта у стан логічної одиниці та ініціює роботу блока *керування передавачем.* Блок керування передавачем через один машинний цикл формує сигнал дозволу «Посилка», який дозволяє видавання вмісту регістра зсуву передавача на вивід *RxD* та сигналу «Синхронізація зсуву» на вивід *TxD.* За сигналом «Зсув» у момент стану *S6P2* кожного машинного циклу вміст регістра зсуву передавача зсувається праворуч на одну позицію молодши-

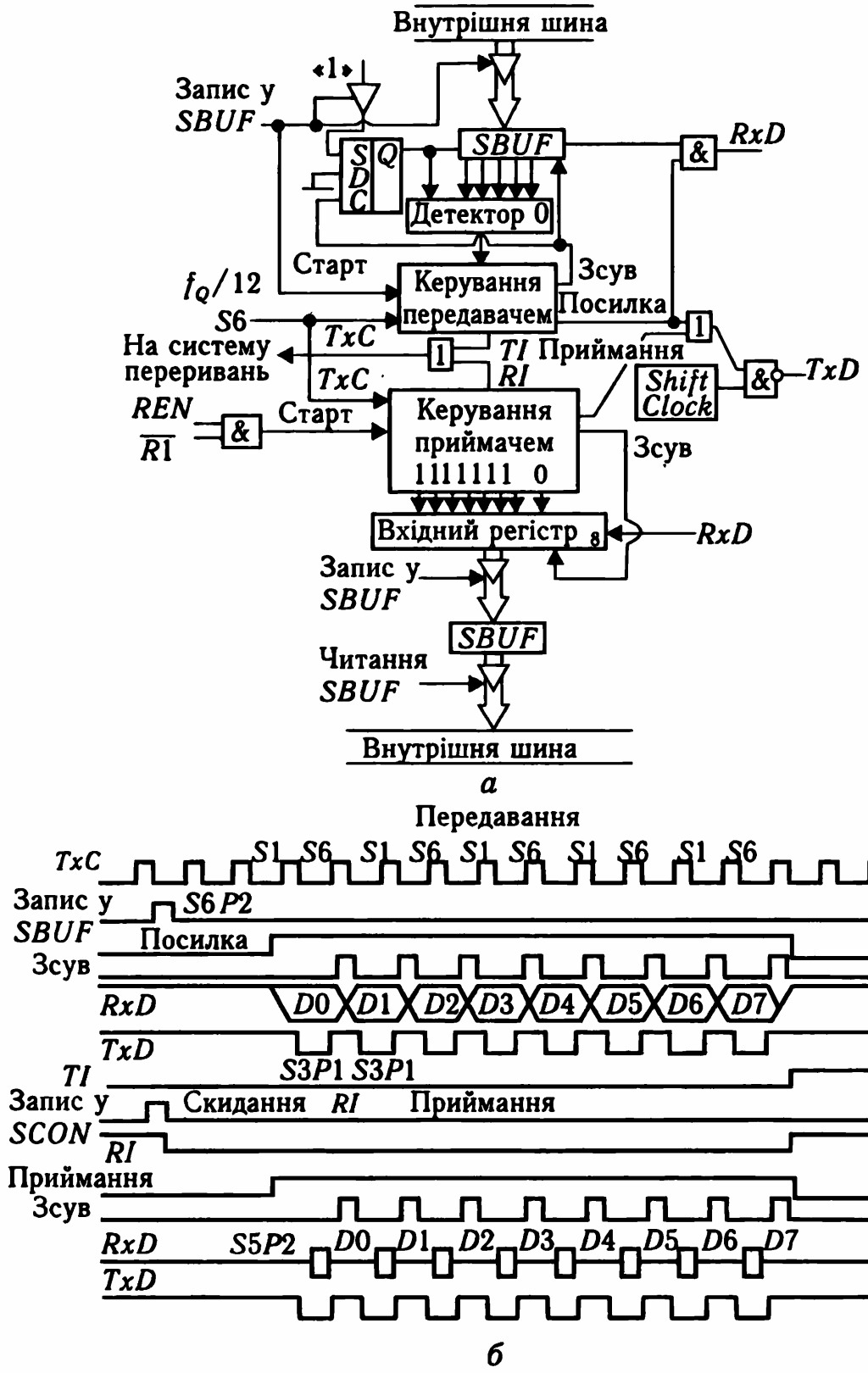


Рис. 6.12. Послідовний порт в режимі 0:
а — функціональна схема; *б* — діаграми

ми бітами вперед і надходить на вивід *RxD*. У старші біти регістра зсуву передавача, що звільняються, записуються нулі.

Під час отримання від *детектора нуля* сигналу «Передавач звільнений» блок керування передавачем знімає сигнал «Посилка» і встановлює прапорець переривання передавача *TI* на початку інтервалу *S1P1* десятого машинного циклу після надходження сигналу «Запис у *SBUF*».

Приймання починається за одночасного виконання умов *REN* = 1 і *RI* = 0. На початку інтервалу *S6P2* наступного машинного циклу (див. рис. 6.11, *б*) блок *керування приймачем* формує сигнал дозволу «Приймання», за яким на вихід *TxD* передаються синхросигнали зсуву, і в *регістрі зсуву приймача* починають формуватися значення бітів даних, які зчитуються з виводу *RxD* в інтервали *S5P2* кожного машинного циклу. В інтервали *S6P2* кожного машинного циклу за сигналом «Зсув» здійснюється зсув вмісту регістра зсуву приймача ліворуч на одну позицію, і прийнятий біт записується у крайній правий розряд. Після надходження восьмого імпульсу «Зсув» вміст регістра приймача переписується у *SBUF*. В інтервалі *S1P1* десятого машинного циклу блок керування приймачем переписує вміст регістра зсуву у буфер *SBUF*, знімає сигнал «Приймання» та встановлює прапорець переривання приймача *RI* у стан логічної одиниці.

У ***режимі 1*** послідовний порт — 8-розрядний універсальний асинхронний приймач-передавач зі змінною швидкістю обміну. Через *TxD* передаються, а через *RxD* приймаються 10 бітів: нульовий старт-біт, 8 біт інформації та одиничний стоп-біт. Швидкість обміну є змінною. Вона визначається частотою переповнення таймера 1 f_{out1} і бітом *SMOD* регістра *PCON*. На рис. 6.13, *а, б* показано відповідно функціональну схему і діаграми послідовного порту в режимі 1.

Передавання починається з будь-якої команди, що використовує *SBUF* як регістр призначення, наприклад:

```
MOV SBUF, #25    ; переслати в SBUF число 25.
```

Пристрій керування ОМК за сигналом «Запис у *SBUF*» завантажує 1 у дев'ятий біт *регістра зсуву передавача*, ініціює роботу блока *керування передавачем* та в інтервал *S1P1* формує сигнал дозволу «Посилка» низького рівня. За цим сигналом на вивід *TxD* спочатку надходить старт-біт, а потім за сигналом дозволу «Дані» за імпульсами «Зсув» вміст регістра передавача зсувається на одну позицію, і по черзі на вивід *TxD* надходять 8 біт даних. За дев'ятим імпульсом «Зсув» формується одиничний стоп-біт, прапорець *TI* вста-

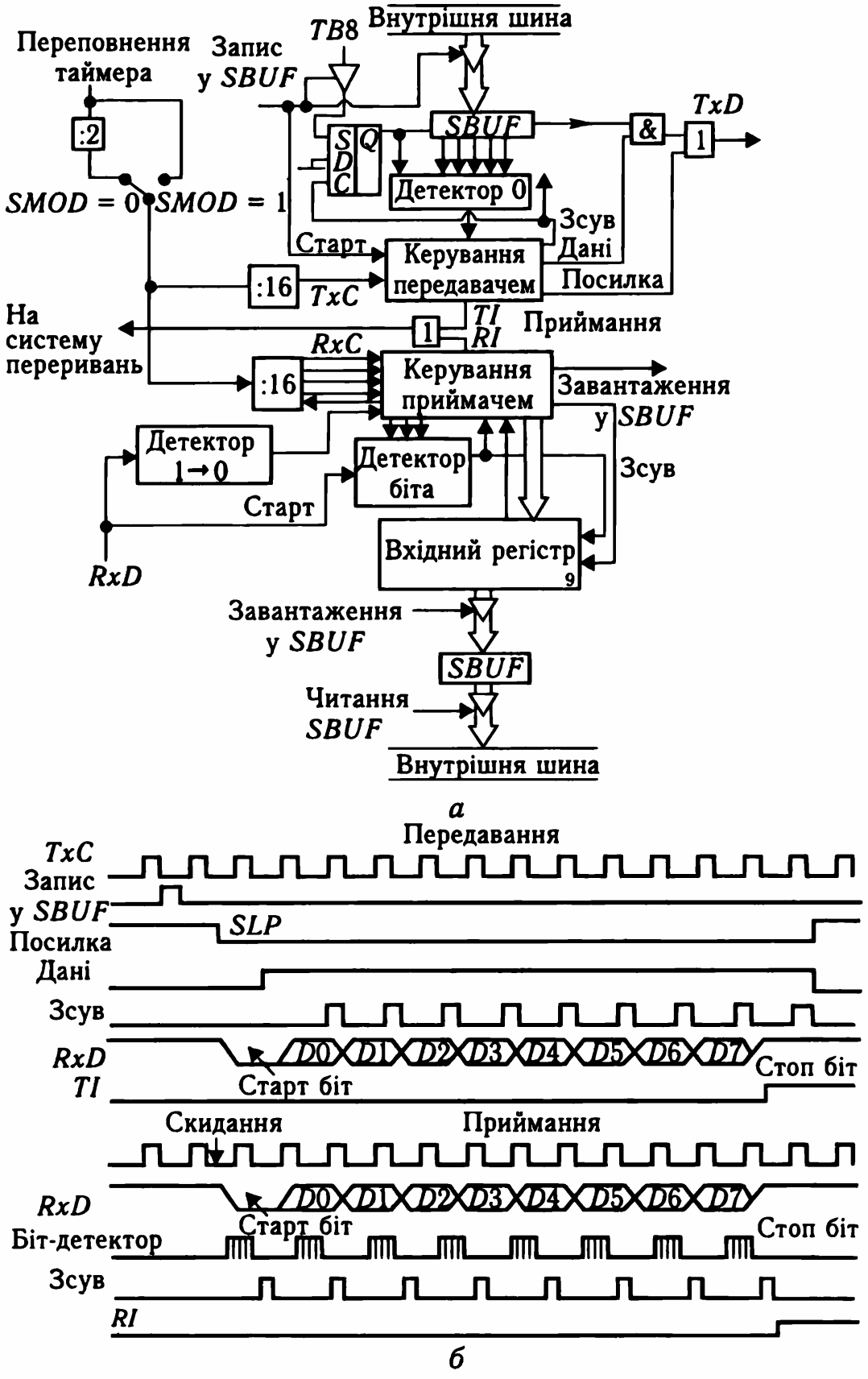


Рис. 6.13. Послідовний порт у режимі 1:
а — функціональна схема; *б* — діаграми

новлюється у стан логічного нуля, сигнали «Посилка» і «Дані» знімаються, і передавання закінчується.

Приймання починається після переходу сигналу на вході *RxD* зі стану «1» у стан «0», який виявляється за допомогою *детектора спадання*. Як тільки перехід зі стану «1» у стан «0» виявлено, у *регістр зсуву приймача* завантажується число *1FFH*, тобто всі 9 розрядів регістра заповнюються одиницями. У процесі переходу сигналу на вході *RxD* зі стану «1» у стан «0» відбувається також скидання значення внутрішнього лічильника-подільника частоти на 16, який формує сигнал «Синхронізація приймача». Внутрішній лічильник починає відлічувати імпульси синхронізації. Під час сьомого, восьмого та дев'ятого імпульсів здійснюється опитування сигналу на вході *RxD* для підтвердження нульового значення старт-біта. Отримані три значення прийнятого біта надходять на *детектор біта*, який визначає дійсне значення прийнятого біта за мажоритарним принципом «два з трьох». При цьому дійсне значення старт-біта дорівнює нулю і починається приймання по черзі 8 біт даних. Значення кожного біта даних також перевіряється детектором біта у сьомому, восьмому та дев'ятому імпульсах сигналу «Синхронізація приймача», і лише після цього заноситься у регістр зсуву приймача. Якщо значення старт-біта не дорівнює нулю, блок керування прийманням знову починає пошук переходів сигналу на вході *RxD* зі стану «1» у стан «0».

Приймання старт-біта та 8 біт даних у кожному машинному циклі супроводжується зсувом вмісту регістра приймача на одну позицію за сигналом «Зсув». Після прийняття старт-біта та 8 бітів даних приймається стоп-біт, значення якого обов'язково має бути одиничним. Отже, після десятого імпульсу «Зсув» у регістрі приймача знаходяться 8 біт інформації і стоп-біт. Блок керування прийманням формує сигнал «Завантаження буфера», за яким вісім інформаційних бітів надходять у *SBUF*, стоп-біт — у розряд *RB8* регістра *SCON*. Прапорець переривання приймача *RI* встановлюється у стан «0». Приймання закінчується, і послідовний порт знову починає процес виявлення переходу сигналу на вході *RxD* зі стану «1» у стан «0».

У ***режимах 2*** і ***3*** послідовний порт — це 9-розрядний універсальний синхронний приймач-передавач з фіксованою (для режиму 2) та змінною (для режиму 3) швидкістю обміну. У режимі 2 швидкість обміну дорівнює $f_Q/32$ за $SMOD = 1$ або $f_Q/64$ за $SMOD = 0$. У режимі 3 швидкість обміну визначається таймером 1, як і в режимі 1.

На рис. 6.14, *а*, *б* показано функціональну схему і діаграми послідовного порту в режимі 2. Функціональна схема

послідовного порту в режимі 3 збігається зі схемою рис. 6.13, *а*, а діаграми — рис. 6.14, *б*.

Через вивід *TxD* послідовний порт передає або з виходу *RxD* приймає 11 біт: нульовий старт-біт, 8 біт даних, програмований дев'ятий біт *TB*8 та одиничний стоп-біт. Режими 2 та 3 відрізняються від режиму 1 лише наявністю дев'ятого програмного біта. Внаслідок цього змінюються умови закінчення циклу приймання: блок керування прийманням формує сигнал керування «Завантаження буфера», завантажує стоп-біт у розряд *RB*8 регістра *SCON* та встановлює прапорець переривання приймача *RI* у стан «1» лише тоді, якщо в останньому такті зсуву виконуються дві умови: $RI = 0$ та $SM2 = 0$ або значення дев'ятого прийнятого біта даних дорівнює одиниці.

Значення стоп-біта у режимах 2 і 3 не впливає на регістри *SBUF*, *RB*8 або *RI*.

Швидкість приймання-передавання даних у послідовному порту за різних режимів роботи. У режимі 0 частота передавання залежить лише від резонансної частоти кварцового резонатора $f_0 = f_Q / 12$. За один машинний цикл послідовний порт передає 1 біт інформації.

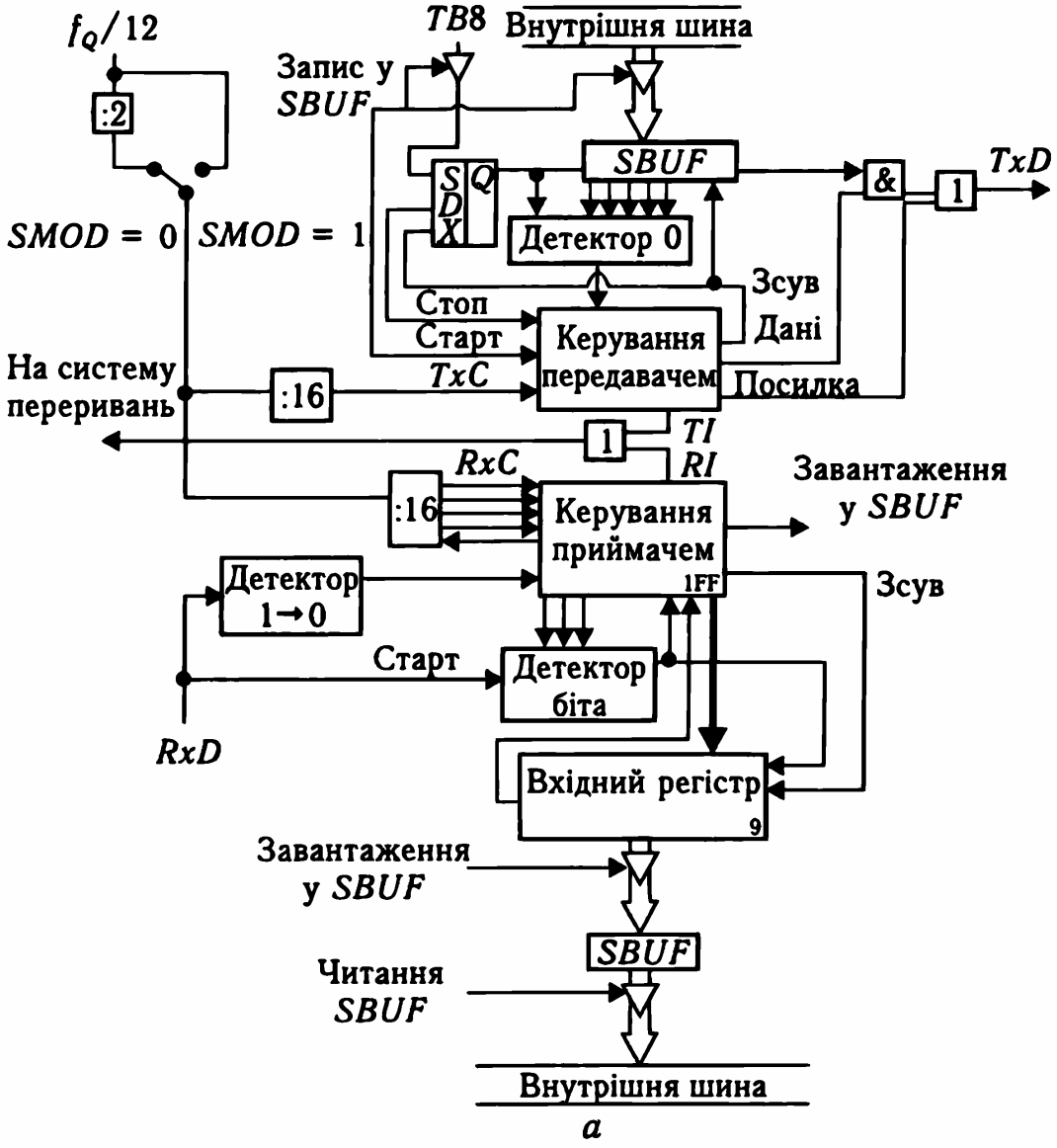
У режимах 1—3 швидкість приймання-передавання залежить від значення керуючого біта *SMOD* (див. табл. 6.2) у регістрі *SCON*. У режимі 2 частота передавання визначається формулою $f_2 = (2^{SMOD} / 64) f_Q$. Отже, якщо $SMOD = 0$, частота передавання дорівнює $(1/64) f_Q$, а якщо $SMOD = 1$, — $(1/32) f_Q$.

У режимах 1 та 3 у формуванні частоти передавання, крім біта *SMOD*, бере участь *таймер-лічильник 1*. При цьому частота передавання залежить від частоти переповнення (*OVT*1) і визначається як $f_{1,3} = (2^{SMOD} / 32) f_{out}$. Переривання від таймера-лічильника 1 у цьому разі має бути заблоковано. Сам таймер-лічильник 1 може працювати і як таймер, і як лічильник подій у будь-якому з трьох режимів. Однак найкраще використовувати режим таймера з автоперевантаженням (старша тетрада $TMOD = 0010B$). При цьому частота передавання визначається як:

$$f_{1,3} = (2^{SMOD} / 32)(f_Q / 12) / (256 - (TH1)),$$

де *TH*1 — вміст регістра *TH*1.

Блок таймерів-лічильників призначений для підрахунку зовнішніх подій (функція лічильника), реалізації програмно керованих затримок та виконання функцій задання часу (функ-



ція таймера). Під час виконання функції таймера вміст T/C інкрементується в кожному машинному циклі, тобто через кожні 12 періодів резонатора. У процесі виконання функції лічильника вміст T/C інкрементується під впливом переходу зі стану «1» у стан «0» зовнішнього вхідного сигналу, що надходить на відповідний ($T0$, $T1$) вивід ОМК. Опитування значення зовнішнього вхідного сигналу виконується у фазі $P2$ стану $S5$ кожного машинного циклу. Вміст лічильника збільшується на одиницю, якщо у попередньому циклі надійшов вхідний сигнал високого рівня (1), а в наступному — сигнал низького рівня (0). Нове інкрементоване значення лічильника формується у фазі $P1$ стану $S3$ машинного циклу, що є наступним після того циклу, в якому був зафіксований перехід зі стану «1» у стан «0». Для фіксування переходу треба два машинних цикли. Тому максимальна частота

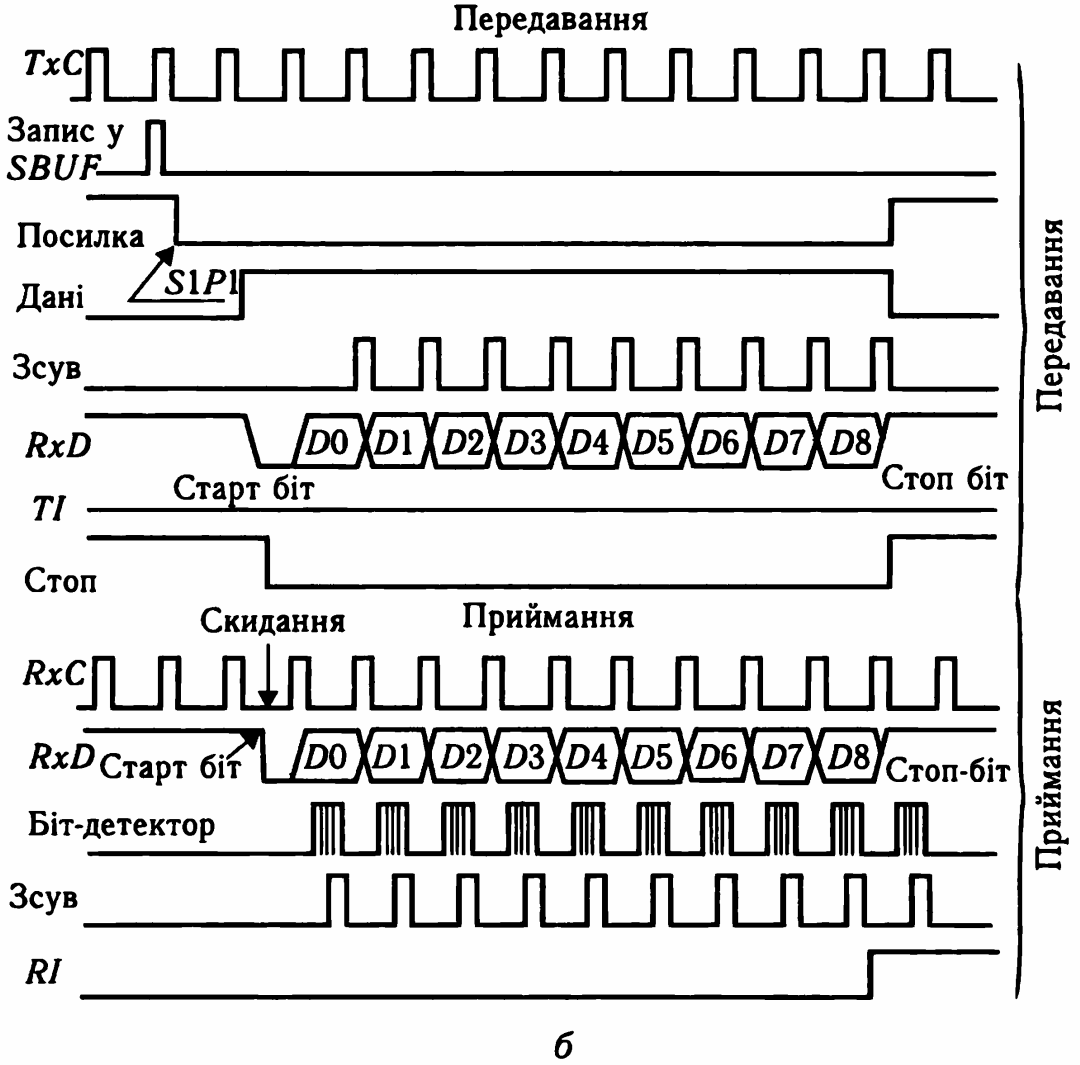


Рис. 6.14. Послідовний порт у режимі 2:
а — функціональна схема; *б* — діаграми

підрахунку вхідних імпульсів дорівнює 1/24 частоти резонатора. Для гарантованого зчитування вхідного сигналу він має утримувати своє значення впродовж щонайменше одиниці машинного циклу ОМК.

До складу блока таймерів-лічильників входять:

- два 16-розрядних регістри $T/C0$ та $T/C1$;
- 8-розрядний регістр режимів *TMOD*;
- 8-розрядний регістр керування *TCON*;
- схема інкремента;
- схема фіксації сигналів $\overline{INT}0$, $\overline{INT}1$, $T0$, $T1$;
- схема керування прапорцями;
- логіка керування T/C.

Регістри $T/C0$ і $T/C1$ виконують функцію зберігання результатів підрахування. Кожний з них складається з двох 8-розрядних регістрів — $TH0$, $TL0$ і $TH1$, $TL1$ відповідно

Таблиця 6.5. **Призначення бітів регістра *TMOD***

Біт	Позначення	Призначення
TMOD.3 для *T/C0* (*TMOD*.7 для *T/C*1)	*GATE*0 (*GATE*1)	Дозволяє керувати таймером від зовнішнього виводу ($\overline{INT0}$ — для T/C0, $\overline{INT1}$ — для *T/C*1) *GATE* = 0 — керування забороняється *GATE* = 1 — керування дозволяється
TMOD.2 для *T/C0* (*TMOD*.6 для *T/C*1)	$C/\overline{T}0$ ($C/\overline{T}1$)	Біт вибору функції таймера або лічильника. Якщо біт скинуто, працює таймер від внутрішнього джерела сигналів синхронізації, а якщо біт встановлено, працює лічильник від зовнішніх сигналів на вході *T*0 (*T*1)
TMOD.1 для *T/C*0 (*TMOD*.5 для *T/C*1)	*M*1.0 (*M*1.1)	Вибір режиму роботи
TMOD.0 для *T/C*0 (*TMOD*.4 для *T/C*1)	*M*0.0 (*M*0.1)	

(*TH* — старші, *TL* — молодші регістри). Кожний з цих регістрів має свою адресу і може бути використаний як РЗП, якщо відповідний таймер не використовується.

Початковий код лічення заноситься у регістри *T/C* програмно. Ознакою закінчення лічення є переповнення регістра *T/C*, тобто перехід його вмісту зі стану «всі одиниці» у стан «усі нулі».

Регістр режимів TMOD призначений для приймання та зберігання коду, який визначає:

- один з чотирьох можливих режимів роботи кожного *T/C*;
- виконання функцій таймерів або лічильників;
- керування *T/C* на зовнішньому виводі.

Призначення бітів регістра *TMOD* наведено у табл. 6.5.

Вибирання режиму роботи здійснюється окремо для *T/C*0 та *T/C*1 згідно зі значеннями бітів *M*1.0 (*M*1.1) і *M*0.0 (*M*0.1):

*M*1	*M*0	Режим
0	0	0
0	1	1
1	0	2
1	1	3

Регістр керування TCON призначений для прийняття і зберігання коду керуючого слова. Призначення бітів регістра *TCON* наведено у табл. 6.6.

Прапорці переповнення *TF*0 і *TF*1 встановлюються апаратно після переповнення відповідних *T/C* (перехід вмісту регістра *T/C* зі стану «всі одиниці» у стан «усі нулі»). Якщо при цьому переривання від відповідного *T/C* дозволяється, то встановлення прапорця *TF* викликає переривання. Скидання прапорців *TF*0 і *TF*1 відбувається апаратно під час передавання керування програмі оброблення відповідного переривання. Переривання прапорців *TF*0 та *TF*1 може відбуватися після виклику (встановлення *TF*) або відміни (скидання *TF*).

Прапорці *IE*0 та *IE*1 встановлюються апаратно від зовнішніх джерел переривань (відповідно входи ОМК $\overline{INT\,0}$ та $\overline{INT\,1}$) або програмно та ініціюють виклик програми обробки відповідного переривання. Скидання цих прапорців виконується апаратно під час обслуговування переривання лише тоді, коли переривання зумовлено фронтом сигналу. Якщо переривання *зумовлено рівнем сигналу на вході* $\overline{INT\,0}$ ($\overline{INT\,1}$), *то скидання прапорця IE* має виконати програма обслуговування переривання, яка впливає на джерело переривання для зняття ним запиту.

Таблиця 6.6. **Призначення бітів регістра *TCON***

Біт	Позначення	Призначення
7 (5)	*TF*1 (*TF*0)	Прапорці переповнення. Їхнє скидання і встановлення відбувається апаратно і програмно. Доступні для читання
6 (4)	*TR*1 (*TR*0)	Біти вимикання окремо для *T/C*0 та *T/C*1: *TR* = = 0 — вимкнений; *TR* = 1 — увімкнений
3 (1)	*IE*1 (*IE*0)	Прапорці запиту зовнішніх переривань на входах $\overline{INT1}$ ($\overline{INT0}$). Скидання і встановлення відбувається апаратно і програмно. Доступні для читання
2 (0)	*IT*1 (*IT*0)	Біти, що визначають тип переривання на входах $\overline{INT1}$ ($\overline{INT0}$): *IT* = 0 — переривання за рівнем (низькому); *IT* = 1 — переривання за фронтом (перехід зі стану «1» у стан «0»)

Примітка. Біти 4, 5 належать до *T/C*0, а біти 6,7 — до *T/C*1. Біти 0, 1 визначають зовнішні переривання на вході $\overline{INT0}$, біти 2,3 — на вході $\overline{INT1}$.

Схема інкремента призначена:

• для збільшення на одиницю у кожному машинному циклі вмісту регістрів $T/C0$, $T/C1$, для яких встановлений режим таймера та дозволено лічення;

• для збільшення на одиницю у циклі вмісту регістрів $T/C0$, $T/C1$, для яких встановлено режим лічильника, дозволено лічення і на відповідному вході ОМК ($T0$ для $T/C0$ та $T1$ для $T/C1$) зафіксовано лічильний імпульс.

Схема фіксації $\overline{INT0}$, $\overline{INT1}$, $T0$, $T1$ — це чотири тригери. У фазі $P2$ стану $S5$ кожного машинного циклу у них запам'ятовується інформація, яка надійшла з виводів $\overline{INT0}$, $\overline{INT1}$, $T0$, $T1$.

Схема керування прапорцями встановлює і скидає прапорці переповнення T/C та прапорці запитів зовнішніх переривань.

Логіка керування синхронізує роботу регістрів $T/C0$ та $T/C1$ згідно із запрограмованими режимами роботи і синхронізує роботу блока T/C з роботою ОМК.

Режими роботи блока T/C. Режим роботи кожного блока T/C визначається значеннями бітів $M0$, $M1$ у регістрі $TMOD$. Таймери $T/C0$ та $T/C1$ мають чотири режими роботи. Режими 0, 1, 2 однакові для обох T/C; у цих режимах вони повністю незалежні один від одного. Робота $T/C0$ та $T/C1$ у режимі 3 неоднакова. При цьому встановлення режиму 3 у таймері $T/C0$ впливає на режим роботи таймера $T/C1$.

Режим 0 ($M0 = 0$, $M1 = 0$). Таймер у режимі 0 — це пристрій на базі 13-розрядного регістра, і він є 8-розрядним таймером (лічильником) з п'ятирозрядним передподільником на 32.

Для $T/C0$ 13-розрядний регістр складається з 8 розрядів регістра $TH0$ і п'яти молодших розрядів регістра $TL0$, а для $T/C1$ — з 8 розрядів регістра $TH1$ і чотирьох молодших розрядів регістра $TL1$. Функцію подільника на 32 виконують регістри $TL0$, $TL1$. Вони є програмно-доступними, але значущими в них є лише п'ять молодших розрядів. Функціональну схему $T/C1$ у режимі 0 (схема $T/C0$ є аналогічною) зображено на рис. 6.15, де OSC — джерело синхронізації ОМК (внутрішнє або зовнішнє). На виході OSC — сигнал з частотою f_Q. Біт $C/\overline{T}$ регістра $TMOD$ визначає виконання функцій таймера ($C/\overline{T} = 0$) або лічильника ($C/\overline{T} = 1$). Лічення починається за командою, яка встановлює біт TR регістра $TCON$ у стан «1», наприклад за командою $SETB$ $TR1$. Якщо треба керувати лічennям ззовні, то біт $GATE$ ре-

гістра $TMOD$ встановлюється у стан «1». Тоді за $TR = 1$ лічення дозволяється, якщо на вході $\overline{INT0}$ (для $T/C0$) або $\overline{INT1}$ (для $T/C1$) встановлено одиничний стан, і заборонено, якщо встановлений нульовий стан. Встановлення бітів $TR0$ (для $T/C0$) і $TR1$ (для $T/C1$) в одиничний стан вимикає відповідний блок T/C незалежно від стану інших бітів.

У разі переповнення T/C, тобто під час переходу вмісту регістра T/C зі стану «всі одиниці» у стан «усі нулі», встановлюється прапорець $TF0$ (для $T/C0$) або $TF1$ (для $T/C1$) у регістрі $TCON$.

Режим 1 ($M0 = 1$, $M1 = 0$). Відмінність від режиму 0 полягає в тому, що встановлення режиму 1 перетворює T/C на пристрій із 16-розрядним регістром. Для $T/C0$ регістр складається з програмно доступних пар $TL0$, $TH0$, для $T/C1$ — з програмно доступних пар $TL1$, $TH1$. Функціональну схему на прикладі $T/C1$ зображено на рис. 6.16.

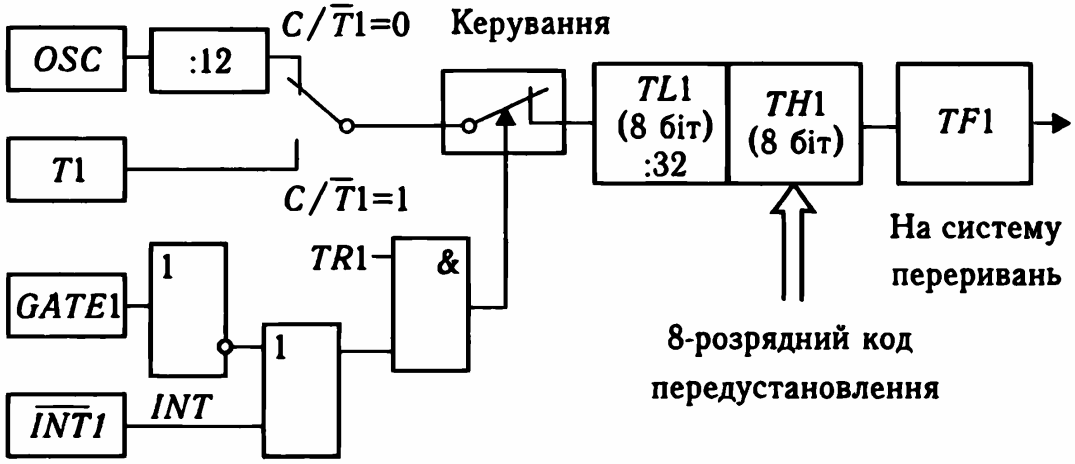


Рис. 6.15. Функціональна схема $T/C1$ у режимі 0

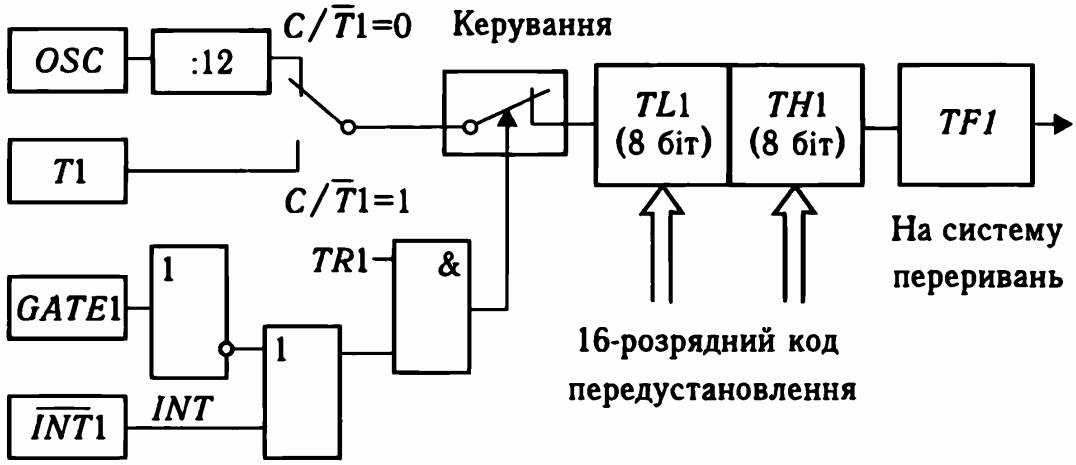


Рис. 6.16. Функціональна схема $T/C1$ у режимі 1

Режим 2 (M0 = 0, M1 = 1). У режимі 2 T/C — це пристрій на базі 8-розрядного регістра $TL0$ для $T/C0$ та регістра $TL1$ для $T/C1$. Під час кожного переповнення TL, крім встановлення в регістрі $TCON$ прапорця TF, відбувається автоматичне перезавантаження вмісту TH у TL. Регістри $TH0$ та $TH1$ завантажуються програмно. Перезавантаження $TL0$ з $TH0$ та $TL1$ з $TH1$ не впливає на вміст регістрів $TH0$ та $TH1$. Функціональну схему $T/C1$ у режимі 2 показано на рис. 6.17.

Режим 3. Таймер-лічильник 1 заблокований і зберігає своє значення. Таймер-лічильник 0 у режимі 3 — це два незалежних пристрої на базі 8-розрядних регістрів $TL0$ і $TH0$ (рис. 6.18). Пристрій на базі $TL0$ може працювати як у режимі таймера, так і в режимі лічильника, а на базі $TH0$ — лише у режимі таймера.

Як видно з рис. 6.18, для забезпечення роботи $T/C0$ у режимі 3 використовуються біти $TR1$ та $TF1$, тому вони не можуть використовуватися для керування $T/C1$. Це призводить до того, що під час встановлення $T/C0$ у режим 3, а $T/C1$ — у режими 0, 1 або 2 — таймер $T/C1$ за $GATE1 = 1$ завжди ввімкнений. Якщо $GATE1 = 0$, вмикання $T/C1$ визначається зовнішніми сигналами аналогічно розглянутому режиму 0. У разі переповнення у режимах 0 і 1 $T/C1$ відбувається встановлення у стан логічного нуля, а в режимі 2 — перезавантажується без встановлення прапорця $TF1$.

Оскільки $T/C1$ апаратно пов'язаний з послідовним портом, то під час роботи $T/C0$ у режимі 3 можна використовувати $T/C1$ у режимі 2 для задання швидкості роботи послідовного порту або для інших задач, що не потребують переривання.

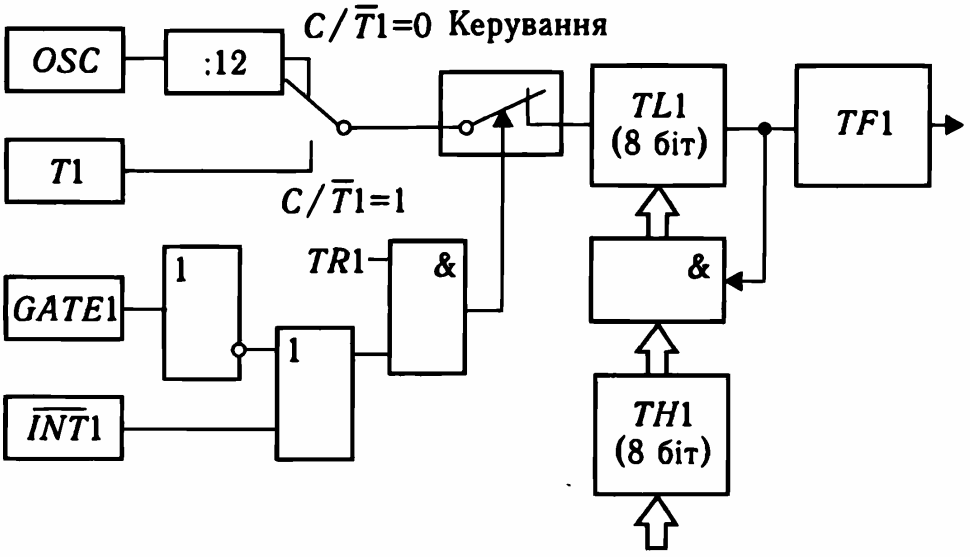


Рис. 6.17. Таймер-лічильник у режимі 2

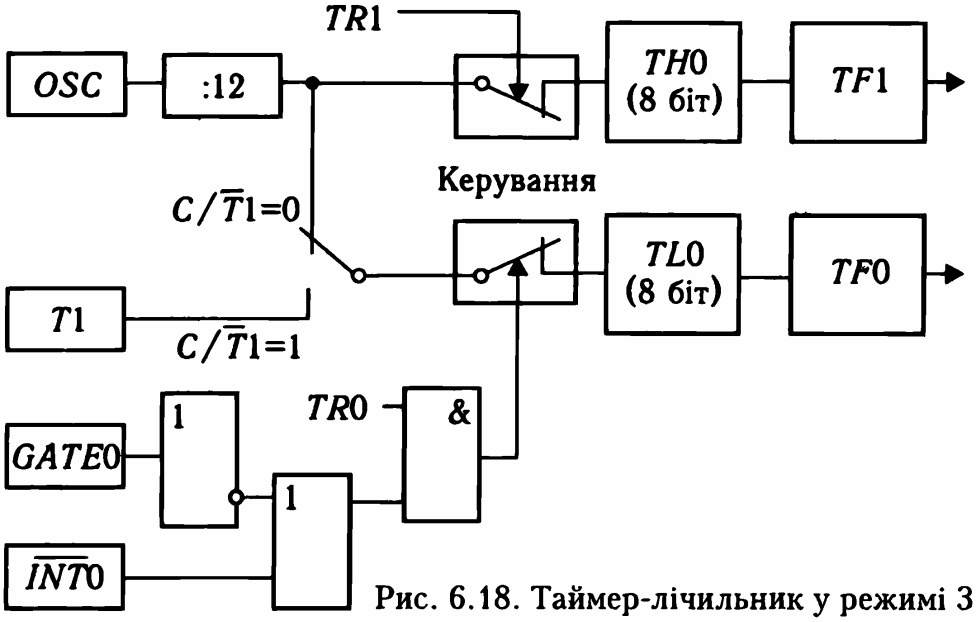


Рис. 6.18. Таймер-лічильник у режимі 3

Система переривань (рис. 6.19) призначена для реагування на зовнішні та внутрішні події. До зовнішніх подій належать появи нульового потенціалу (або зрізу) на виводах $\overline{INT\,0}$, $\overline{INT\,1}$, до внутрішніх — переповнення таймерів-лічильників, завершення послідовного обміну. Зовнішні або внутрішні події викликають встановлення відповідних прапорців: *IE*0, *IE*1, *TF*0, *TF*1, *R*I і *T*I, що й спричиняють переривання. Зазначимо, що всі перелічені прапорці можуть бути програмно встановлені або скинуті, при цьому їхнє програмне встановлення викликає переривання так само, як і реагування на подію. Отже, переривання можуть програмно викликатися або програмно усуватися. Крім того, переривання на входах $\overline{INT\,0}$, $\overline{INT\,1}$ можуть викликатися програмним скиданням бітів *P*3.2 і *P*3.3. Керування системою переривання здійснюється за допомогою запису керуючих слів у регістри *TCON* (див. табл. 6.6), *IE* і *IP*. Регістр дозволу переривань *IE* призначений для дозволу або заборони переривань від відповідних джерел. Регістр пріоритетів переривань *IP* використовують для встановлення рівня пріоритету переривання для кожного з п'яти джерел переривань.

Призначення бітів регістрів *IE* та *IP* наведено відповідно у табл. 6.7 та 6.8.

Зовнішні переривання залежно від стану бітів *IT*0, *IT*1 регістра *TCON* (див. табл. 6.6) сприймаються або за переходом сигналу на входах $\overline{INT\,0}$ та $\overline{INT\,1}$ з *H*-рівня у *L*-рівень, або за нульовим рівнем сигналу. Під час переривання за нульовим рівнем цей рівень має утримуватися не менше ніж 12 періодів сигналу тактової частоти *CLK*. Після надходження

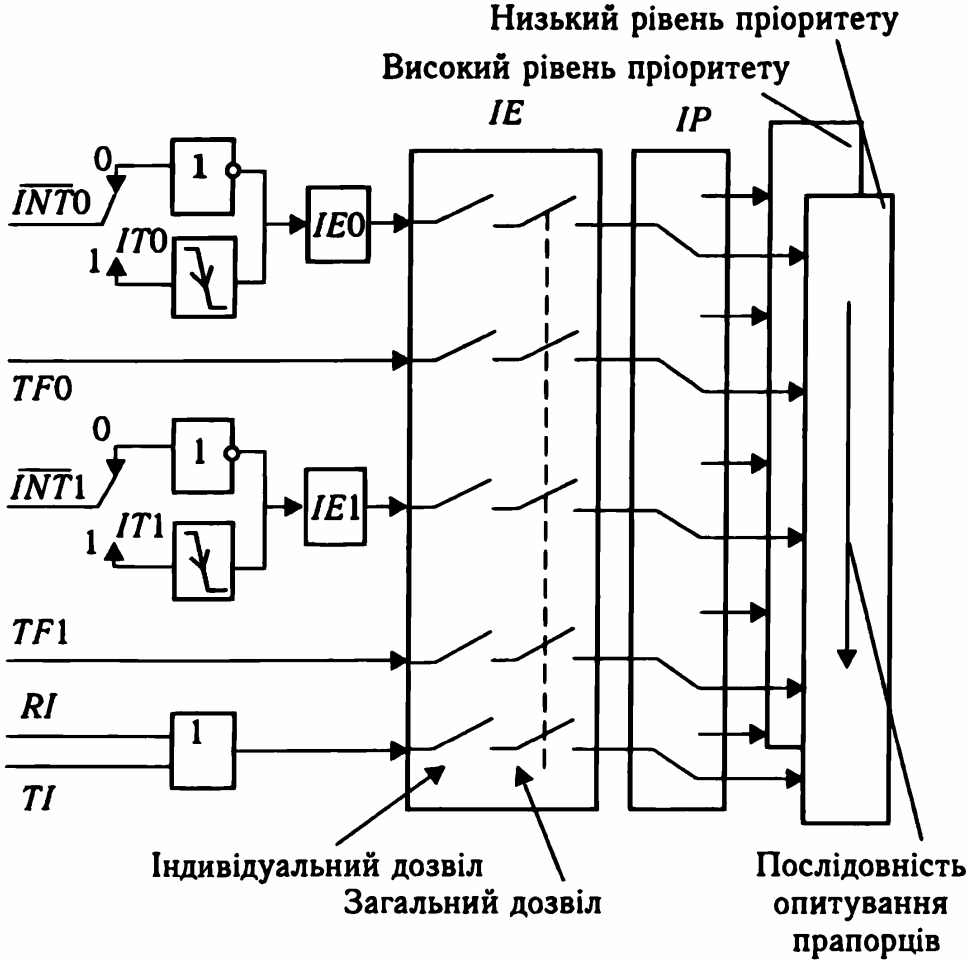


Рис. 6.19. Функціональна схема системи переривань

одного із сигналів $\overline{INT\,0}$ або $\overline{INT\,1}$ встановлюється прапорець $IE0$ або $IE1$ у регістрі $TCON$, що викликає відповідне переривання.

Скидання прапорців $IE0$ або $IE1$ здійснюється апаратно лише тоді, коли переривання здійснюється за переходом зі стану «1» у стан «0» сигналу.

Якщо переривання викликано нульовим рівнем сигналу, то скиданням прапорців $IE0$ або $IE1$ керує відповідна підпрограма обслуговування переривання через вплив на джерело переривання з метою зняття ним запиту.

Переривання від таймерів-лічильників викликаються одиничними значеннями прапорців $TF0$ або $TF1$ у регістрі $TCON$. Прапорці $TF0$ та $TF1$ встановлюються під час переповнення відповідних таймерів. Скидання прапорців $TF0$ та $TF1$ відбувається автоматично у процесі переходу до підпрограм оброблення переривань.

Переривання від послідовного порту викликаються встановленням прапорців TI або RI у регістрі $SCON$. Скидання

Таблиця 6.7. **Призначення бітів регістра *IE***

Біт	Позначення	Призначення
IE.7	*EA*	Зняття блокування переривань. Скидання здійснюється програмно для заборони всіх переривань незалежно від стану *IE*4 – *IE*0
IE.6	–	Не використовуються
IE.5	–	
IE.4	*ES*	Біт дозволу переривання від універсального синхронного приймача-передавача. Встановлення-скидання програмою для дозволу-заборони переривань від прапорців *TI* або *RI*
IE.3	*ET*1	Біт дозволу переривання від таймера 1. Встановлення-скидання програмою для дозволу-заборони переривань від таймера 1
IE.2	*EX*1	Біт дозволу зовнішнього переривання 1. Встановлення-скидання програмою для дозволу-заборони переривань
IE.1	*ET*0	Біт дозволу переривання від таймера 0. Працює аналогічно *IE*.3
IE.0	*EX*0	Біт дозволу зовнішнього переривання 0. Працює аналогічно *IE*.2

Таблиця 6.8. **Призначення бітів регістра *IP***

Біт	Позначення	Призначення
IP.7 – *IP*.5	–	Не використовуються
IP.4	*PS*	Біт пріоритету універсального асинхронного приймача-передавача. Встановлення-скидання програмою для присвоєння перериванню від універсального асинхронного приймача-передавача вищого-нижчого пріоритету
IP.3	*PT*1	Біт пріоритету таймера 1. Встановлення-скидання програмою для присвоєння перериванню від таймера 1 вищого-нижчого пріоритету
IP.2	*PX*1	Біт пріоритету зовнішнього переривання 1. Установлення-скидання програмою для присвоєння вищого-нижчого пріоритету перериванню $\overline{INT1}$
IP.1	*PT*0	Біт пріоритету таймера 0. Працює аналогічно *IP*.3
IH.0	*PX*0	Біт пріоритету зовнішнього переривання 0. Працює аналогічно *IP*.2

прапорців *TI* або *RI* здебільшого здійснюється у підпрограмі оброблення переривання.

Кожний з описаних типів переривань може бути дозволений або заборонений за допомогою встановлення-скидання відповідного біта у регістрі *IE* (див. табл. 6.7). Скиданням біта *EA* можна заборонити одночасно всі переривання.

До складу системи переривань входять також логіка оброблення прапорців переривань та схема формування вектора переривання. *Логіка оброблення прапорців переривань* здійснює пріоритетний вибір запиту переривання, скидання відповідного прапорця та ініціює апаратну реалізацію команди переходу на підпрограму обслуговування переривання. Кожному з джерел переривань за допомогою встановлення-скидання відповідного біта у регістрі *IP* (див. табл. 6.8) присвоюють один з двох рівнів пріоритету — високий або низький. Програма оброблення переривання може перериватися іншим запитом переривання того самого рівня пріоритету. Програма оброблення, яка має низький рівень переривань, може бути перервана запитом переривання з високим рівнем. За одночасного надходження запитів з різними рівнями спочатку обслуговується запит з високим рівнем пріоритету; за одночасного надходження запитів з однаковими рівнями оброблення їх здійснюється у порядку послідовності внутрішнього опитування прапорців (див. рис. 6.19, напрям показано стрілкою).

Схема формування вектора переривання залежно від джерела переривання формує двобайтові адреси підпрограм обслуговування переривання (табл. 6.9).

Режими енергоспоживання ОМК. В ОМК, виконаних за *n*-МДН-технологією, регістр *PCON* має лише 1 біт *SMOD*, що керує швидкістю передавання послідовного порту. Тому існує лише *режим зниженого споживання*, який забезпечує живлення внутрішнього ОЗП, якщо значення сигналу на виводі $\overline{RST}$ більше, ніж на виводі U_{CC}. Це реалізується за до-

Таблиця 6.9. **Вектори переривання**

Джерело переривання	Вектор переривання
Зовнішнє переривання $\overline{INT0}$	0003*H*
Таймер-лічильник *T/C*0	000*BH*
Зовнішнє переривання $\overline{INT1}$	0013*H*
Таймер-лічильник *T/C*1	001*BH*
Послідовний порт	0023*H*

помогою двох діодів, з катодів яких здійснюється живлення ОЗП, а аноди з'єднані з виводами *RST* та U_{CC} (рис. 6.20).

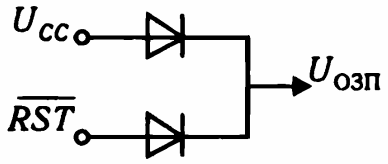


Рис. 6.20. Забезпечення живлення ОЗП у режимі зниженого споживання

В ОМК, виконаних за КМДН-технологією, є два режими зменшеного енергоспоживання: *режим холостого ходу* і *режим мікроспоживання*. Джерелом живлення у режимі холостого ходу та мікроспоживання є вивід U_{CC}. Вибір і керування режимами здійснюється за допомогою регістра керування споживанням *PCON*. Адресація окремих бітів у регістрі *PCON* недопустима. Формат регістра *PCON* наведено у табл. 6.10.

Режими зменшеного енергоспоживання ініціюються встановленням бітів *PD* та *IDL*. Вплив значення цих бітів на формування тактових сигналів блоків ОМК показано на рис. 6.21.

Режим холостого ходу задається командою, яка встановлює біт *IDL* у стан «1», наприклад, *MOV PCON*, #01. У цьому режимі блокуються функціональні вузли блока ЦП (*CPU*). Внутрішній генератор сигналів синхронізації продовжує ро-

Таблиця 6.10. **Позначення бітів регістра *PCON***

Біт	Позначення	Призначення	Примітка
7	*SMOD* (*Serial MODe*)	Біт подвоєння швидкості передавання	Керує роботою послідовного порту. Якщо *SMOD* = = 1, швидкість передавання подвоюється у режимах 1, 2, 3 послідовного порту
6	—	Резервний	
5	—	—«—	
4	—	—«—	
3	*GF*1 (*General purpose Flag bit*)	Прапорець загального призначення	
2	*GF*2	Те саме	
1	*PD* (*Power Down bit*)	Біт вмикання режиму мікроспоживання	Лише для КМДН-технологій. Якщо біти *PD* і *IDL* одночасно дорівнюють одиниці, перевагу має біт *PD*, тобто автоматично обирається режим мікроспоживання
0	*IDL* (*IDLe mode bit*)	Біт встановлення режиму холостого ходу	

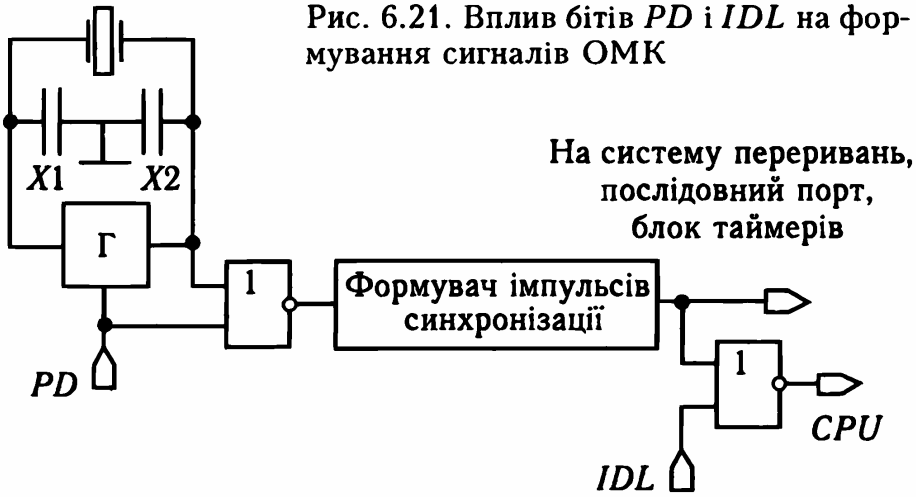


Рис. 6.21. Вплив бітів *PD* і *IDL* на формування сигналів ОМК

боту. Всі регістри, покажчик стеку, програмований лічильник, *PSW*, акумулятор та внутрішній ОЗП зберігають свої значення. На виводах усіх портів утримується той логічний стан, який був на них у момент переходу в режим холостого ходу. На виводах *ALE* та $\overline{PME}$ формується рівень логічної одиниці.

Існує два способи виходу з режиму холостого ходу — за перериванням або за сигналом апаратного скидання на вході *RST*. Активізація будь-якого дозволеного переривання автоматично веде до встановлення біта *IDL* у стан «0», тобто до припинення режиму холостого ходу. Після виконання команди *RETI* (вихід з підпрограми обслуговування переривання) виконується команда, наступна за командою, що перевела ОМК у режим холостого ходу.

Закінчення режиму холостого ходу відбувається також з появою сигналу апаратного скидання на виводі $\overline{RST}$ тривалістю не менше двох машинних циклів. Активний сигнал на виводі $\overline{RST}$ асинхронно скидає біт *IDL* у стан логічного нуля. Оскільки тактовий генератор працює, ОМК відразу після скидання біта *IDL* у стан логічного нуля починає виконувати програму з команди, наступної після команди, що викликала режим холостого ходу. Тривалість інтервалу між скиданням біта *IDL* та моментом, коли вмикається внутрішній алгоритм скидання, може становити до двох машинних циклів. Упродовж цього інтервалу блокується доступ до внутрішнього ОЗП, але не блокується доступ до портів. Тому використовувати команди звернення до портів безпосередньо після команди встановлення біта *IDL* не рекомендується.

Біти *GF*0 та *GF*1 (див. табл. 6.10) краще використовувати для визначення того режиму, в якому відбувся виклик програми оброблення переривання: нормального режиму або режиму холостого ходу. Наприклад, команда, що викликає

режим холостого ходу, може також встановлювати один або кілька прапорців (*GF*0, *GF*1 або будь-які інші). Програма оброблення переривання, перевіряючи ці прапорці, може визначати режим, в якому відбулося це переривання.

Режим мікроспоживання ініціюється встановленням біта *PD* у стан «1», наприклад, за командою *MOV PCON*, #02. У цьому режимі генератор вимикається, припиняючи роботу всіх вузлів ОМК, зберігається лише вміст ОЗП та регістрів спеціальних функцій. На виводах портів утримуються значення, які відповідають вмісту їхніх буферних регістрів. Виходи сигналів *ALE* та $\overline{PME}$ скидаються. Електроживлення відбувається через вивід $\overline{RST}/V_{PD}$. У цьому режимі напруга U_{CC} може бути зменшена до 2 В і має відновитися до номінального значення перед виходом з режиму. Вийти з режиму мікроспоживання можна лише за сигналом апаратного скидання на виводі $\overline{RST}$ тривалістю не менше ніж 10 мс (тривалість відновлення роботи генератора імпульсів синхронізації). За одночасного значення *IDL* = 1 та *PD* = 1 перевагу має біт *PD*.

6.2. Система команд

Система команд ОМК К1816ВЕ51 містить 111 команд, які поділяють на п'ять груп: команди пересилання; арифметичні команди; логічні команди; команди передавання керування; команди операцій з бітами.

Більшість команд (94) мають формат 1—2 байт і виконуються за один-два цикли (за тактової частоти 12 МГц тривалість циклу дорівнює 1 мкс).

В ОМК використовуються такі типи адресації — пряма, безпосередня і непряма. Систему команд ОМК К1816ВЕ51 наведено у табл. 6.11, де використано такі позначення:

R — один з регістрів *R*0—*R*7;

r — регістр *R*0 або *R*1;

ad — байт з РПД за прямою адресацією (у команді вказується або адреса байта — 00—*FF*, або позначення одного з регістрів спеціальних функцій *SFR*, наприклад, *TH*0, *P*1, *TCON*, *SBUF*);

data — безпосередні 8-розрядні дані (перед числовими значеннями ставлять знак #, наприклад #34);

*d*16 — безпосередні 16-розрядні дані;

add — байт-приймач з прямою адресацією;

ads — байт-джерело з прямою адресацією;

*ad*16 — 16-розрядна адреса;

*ad*11 — 11-розрядна адреса;

*ad*8 — 8-розрядна адреса-зміщення;

Mi(..) — комірка пам'яті (у дужках указано адресу. Індекс зазначає тип пам'яті: РПД — резидентна пам'ять даних; ЗПД — зовнішня пам'ять даних; ПП — пам'ять програм);

bit — біт з прямою адресацією (у команді вказується або адреса біта — 00—*FF*, або розташування його у регістр спеціальних функцій, наприклад *ACC*. 0 — нульовий біт акумулятора, *P*1. 2 — другий біт порту 1 або позначення біта у регістр спеціальних функцій, наприклад *EA* — біт загального дозволу переривань).

Вплив команд на прапорці ілюструє табл. 6.12, у якій позначено: «+» — команда впливає на прапорець, «−» — не впливає; «1» — встановлює у стан «1»; «0» — скидає у стан «0».

Таблиця 6.11. **Система команд ОМК К1816ВЕ51**

Мнемокод	Опис	Алгоритм	Байт	Цикл
	КОМАНДИ ПЕРЕСИЛАННЯ			
MOV A, R	Пересилання даних з регістра в акумулятор	$A \leftarrow R$	1	1
MOV A, ad	Пересилання в акумулятор байта з прямою адресацією	$A \leftarrow ad$	2	1
MOV A, @r	Пересилання байта з РПД в акумулятор	$A \leftarrow M_{РПД}\ (r)$	1	1
MOV A, #data	Пересилання безпосереднього операнда в *A*	$A \leftarrow$ *#data*	2	1
MOV R, A	Пересилання акумулятора в регістр	$R \leftarrow A$	1	1
MOV R, ad	Пересилання в регістр байта з прямою адресацією	$R \leftarrow ad$	2	2
MOV R, #data	Пересилання безпосереднього операнда в регістр	$R \leftarrow$ *#data*	2	1
MOV ad, A	Пересилання акумулятора з прямою адресацією	$Ad \leftarrow A$	2	1
MOV ad, R	Пересилання регістра з прямою адресацією	$Ad \leftarrow R$	2	2
MOV add, ads	Пересилання байта з прямою адресацією	$add \leftarrow ads$	3	2
MOV ad, @r	Пересилання байта з РПД з прямою адресацією	$Ad \leftarrow M_{РПД}\ (r)$	2	2
MOV ad, #data	Пересилання безпосереднього операнда з прямою адресацією	$Ad \leftarrow$ *#data*	3	2
MOV @r, A	Пересилання акумулятора в РПД	$M_{РПД}\ (r) \leftarrow A$	1	1

Продовж. табл. 6.11

Мнемокод	Опис	Алгоритм	Байт	Цикл
MOV @r, ad	Пересилання в РПД операнда з прямою адресацією	$M_{\text{РПД}}\ (r) \leftarrow ad$	2	2
MOV @r, #data	Пересилання безпосереднього операнда в РПД	$M_{\text{РПД}}\ (r) \leftarrow$ $\leftarrow$ *#data*	2	1
MOV DPTR, #d16	Завантаження покажчика даних	$DPTR \leftarrow$ *#d*16	3	2
MOVC A, @A+DPTR	Пересилання в акумулятор байта з пам'яті програм	$A \leftarrow M_{\text{ПП}}$ $(A + DPTR)$	1	2
MOVX A, @r	Пересилання байтів із ЗПД в акумулятор	$A \leftarrow M_{\text{ЗПД}}\ (r)$	1	2
MOVX A, @DPTR	Пересилання байта з розширеної ЗПД в акумулятор	$A \leftarrow M_{\text{ЗПД}}$ $(DPTR)$	1	2
MOVX @r, A	Пересилання акумулятора в ЗПД	$M_{\text{ЗПД}}\ (r) \leftarrow A$	1	2
MOVX @DPTR, A	Пересилання байта в розширену ЗПД з акумулятора	$M_{\text{ЗПД}}(DPTR) \leftarrow$ $\leftarrow A$	1	2
PUSH ad	Завантаження у стек	$SP \leftarrow SP + 1;$ $M_{\text{РПД}}(SP) \leftarrow ad$	2	2
POP ad	Витягування зі стеку	$Ad \leftarrow M_{\text{РПД}}(SP);$ $SP \leftarrow SP - 1$	2	2
XCH A, R	Обмін акумулятора і регістра	$A \longleftrightarrow R$	1	1
XCH A, ad	Обмін акумулятора і байта з прямою адресацією	$A \longleftrightarrow ad$	1	1
XCH A, @r	Обмін акумулятора і комірки РПД	$A \longleftrightarrow M$РПД(r)	1	1
XCHD A, @r	Обмін молодших тетрад акумулятора і комірки РПД	$A_{0-3} \longleftrightarrow M_{\text{РПД}}(r)_{0-3}$	1	1
SWAP A	Обмін тетрад в акумуляторі	$A_{0-3} \longleftrightarrow A_{4-7}$	1	1
	АРИФМЕТИЧНІ КОМАНДИ			
ADD A, R	Додавання регістра *R* і акумулятора	$A \leftarrow A + R$	1	1
ADD A, ad	Додавання акумулятора і байта з прямою адресацією	$A \leftarrow A + ad$	2	1
ADD A, @r	Додавання байта з РПД і акумулятора	$A \leftarrow A + M_{\text{РПД}}\ (r)$	1	1

Продовж. табл. 6.11

Мнемокод	Опис	Алгоритм	Байт	Цикл
ADD A, #data	Додавання константи з акумулятором	$A \leftarrow A + \#data$	2	1
ADDC A, R	Додавання *R* з акумулятором і прапорця перенесення *C*	$A \leftarrow A + R + C$	1	1
ADDC A, ad	Додавання акумулятора і з прямою адресацією байта з прапорцем	$A \leftarrow A + ad + C$	2	1
ADDC A, @r	Додавання байта з РПД з акумулятором і прапорцем	$A \leftarrow A + M_{РПД}(r) + C$	1	1
ADDC A, #data	Додавання константи з акумулятором і прапорцем С	$A \leftarrow A + \#data + C$	2	1
DA A	Десяткова корекція *A* під час додавання	Алгоритм десяткової корекції	1	1
SUBB A, R	Віднімання з акумулятора регістра і прапорця *C*	$A \leftarrow A - R - C$	1	1
SUBB A, ad	Віднімання з *A* байта з прямою адресацією і прапорця *C*	$A \leftarrow A - ad - C$	2	1
SUBB A, @r	Віднімання з *A* байта РПД і прапорця *C*	$A \leftarrow A - M_{РПД}(r) - C$	1	1
SUBB A, #data	Віднімання з *A* константи і прапорця *C*	$A \leftarrow A - \#data - C$	2	1
INC A	Інкремент *A*	$A \leftarrow A + 1$	1	1
INC R	Інкремент регістра	$R \leftarrow R + 1$	1	1
INC ad	Інкремент байта з прямою адресацією	$ad \leftarrow ad + 1$	2	1
INC @r	Інкремент байта в РПД	$M_{РПД}(r) \leftarrow \leftarrow M_{РПД}(r) + 1$	1	1
INC DPTR	Інкремент покажчика даних	$DPTR \leftarrow DPTR + 1$	1	2
DEC A	Декремент *A*	$A \leftarrow A - 1$	1	1
DEC R	Декремент регістра	$R \leftarrow R - 1$	1	1
DEC ad	Декремент байта з прямою адресацією	$ad \leftarrow ad - 1$	2	1
DEC @r	Дскремент байта в РПД	$M_{РПД}(r) \leftarrow \leftarrow M$РПД $(r) - 1$	1	1
MUL AB	Множення *A* на регістр *B*	$(B, A) \leftarrow A \times B$	1	4
DIV AB	Ділення *A* на регістр *B*	$A \leftarrow A/B$, залишок у *B*	1	4

Продовж. табл. 6.11

Мнемокод	Опис	Алгоритм	Байт	Цикл
	ЛОГІЧНІ КОМАНДИ			
ANL A, R	Логічна операція регістра I і *A*	$A \leftarrow A \wedge R$	1	1
ANL A, ad	*I* байта з прямою адресацією і A	$A \leftarrow A \wedge ad$	2	1
ANL A, @r	*I* байта РПД і *A*	$A \leftarrow A \wedge M_{РПД}(r)$	1	1
ANL A, #data	*I* константи і *A*	$A \leftarrow A \wedge$ #data	2	1
ANL ad, A	*I* байта з прямою адресацією і *A*	$Ad \leftarrow ad \wedge A$	2	1
ANL ad, #data	*I* байта з прямою адресацією і константи	$ad \leftarrow ad \wedge$ #data	3	2
ORL A, R	АБО регістра і *A*	$A \leftarrow A \vee R$	1	1
ORL A, ad	АБО байта з прямою адресацією і *A*	$A \leftarrow A \vee ad$	2	2
ORL A, @r	АБО байта РПД і *A*	$A \leftarrow A \vee M_{РПД}(r)$	1	1
ORL A, #data	АБО константи і *A*	$A \leftarrow A \vee$ #data	2	1
ORL ad, A	АБО байта з прямою адресацією і *A*	$Ad \leftarrow ad \vee A$	2	1
ORL ad, #data	АБО байта з прямою адресацією і константи *A*	$d \leftarrow ad \vee$ #data	3	2
XRL A, R	ВИКЛЮЧАЛЬНЕ АБО регістра і *A*	$A \leftarrow A \oplus R$	1	1
XRL A, ad	ВИКЛЮЧАЛЬНЕ АБО байта з прямою адресацією і *A*	$A \leftarrow A \oplus ad$	2	1
XRL A, @r	ВИКЛЮЧАЛЬНЕ АБО байта РПД і *A*	$A \leftarrow A \oplus M_{РПД}(r)$	1	1
XRL A, #data	ВИКЛЮЧАЛЬНЕ АБО константи і *A*	$A \leftarrow A \oplus$ #data	2	1
XRL ad, A	ВИКЛЮЧАЛЬНЕ АБО байта з прямою адресацією і *A*	$Ad \leftarrow ad \oplus A$	2	1
XRL ad, #data	ВИКЛЮЧАЛЬНЕ АБО байта з прямою адресацією і константи	$Ad \leftarrow ad \oplus$ #data	3	2

Продовж. табл. 6.11

Мнемокод	Опис	Алгоритм	Байт	Цикл
CLR A	Скидання *A*	$A \leftarrow 0$	1	1
CPL A	Інверсія *A*	$A \leftarrow \bar{A}$	1	1
RL A	Циклічний зсув ліворуч	$A_{n+1} \leftarrow A_n n = 0–6$, $A0 \leftarrow A_7$	1	1
RLC A	Зсув ліворуч через прапорець *C*	$A_{n+1} \leftarrow A_n n = 0–6$, $A0 \leftarrow C$, $C \leftarrow A_7$	1	1
RR A	Циклічний зсув праворуч	$A_n \leftarrow A_{n+1}$, $n = 0–6$, $A7 \leftarrow A0$	1	1
RRC A	Зсув праворуч через прапорець *C*	$A_n \leftarrow An_{n+1}$, $n = 0–6$, $A7 \leftarrow C$, $C \leftarrow A0$	1	1
	КОМАНДИ ОПЕРАЦІЙ З БІТАМИ			
CLR C	Скидання прапорця *C*	$C \leftarrow 0$	1	1
CPL C	Інверсія прапорця *C*	$C \leftarrow \bar{C}$	1	1
SETB C	Встановлення прапорця *C*	$C \leftarrow 1$	1	1
CLR bit	Скидання біта	$Bit \leftarrow 0$	2	1
CPL bit	Інверсія біта	$Bit \leftarrow \overline{bit}$	2	1
SETB bit	Встановлення біта	$Bit \leftarrow 1$	2	1
ANL C, bit	ЛОГІЧНЕ І біта і прапорця *C*	$C \leftarrow C \leftarrow bit$	2	2
ANL C,/bit	ЛОГІЧНЕ І інверсії біта і прапорця *C*	$C \leftarrow C \leftarrow \Lambda\, \overline{bit}$	2	2
ORL C, bit	ЛОГІЧНЕ АБО біта і прапорця *C*	$C \leftarrow CV\ bit$	2	2
ORL C,/bit	ЛОГІЧНЕ АБО інверсії біта і прапорця *C*	$C \leftarrow CV\ \overline{bit}$	2	2
MOV C, bit	Пересилання біта у прапорець *C*	$C \leftarrow bit$	2	1
MOV bit, C	Пересилання прапорця *C* у біт	$bit \leftarrow C$	2	2
*LJMP ad*16	Безумовний довгий перехід у повному обсязі пам'яті програм	$PC \leftarrow ad16$	3	2
*AJMP ad*11	Абсолютний перехід усередині сторінки 2 Кбайт	$PC \leftarrow PC + 2$ $PC_{0-10} \leftarrow ad11$	2	2
*SJMP ad*8	Короткий відносний перехід усередині сторінки ПП 256 б	$PC \leftarrow PC + 2$ $PC \leftarrow PC + ad8$	2	2

Продовж. табл. 6.11

Мнемокод	Опис	Алгоритм	Байт	Цикл
	КОМАНДИ ПЕРЕХОДІВ			
JMP @A+DPTR	Побічний відносний перехід	$PC \leftarrow A + DPTR$	1	2
JC ad8	Перехід, якщо $C = 1$	$PC \leftarrow PC + 2$, якщо $C = 1$, то $PC \leftarrow PC + ad8$	2	2
JNC ad8	Перехід, якщо $C = 0$	$PC \leftarrow PC + 2$, якщо $C = 0$, то $PC \leftarrow PC + ad8$	2	2
JZ ad8	Перехід, якщо $A = 0$	$PC \leftarrow PC + 2$, якщо $A = 0$, то $PC \leftarrow PC + ad8$	2	2
JNZ ad8	Перехід, якщо $A \neq 0$	$PC \leftarrow PC + 2$, якщо $A \neq 0$, то $PC \leftarrow PC + ad8$	2	2
JB bit, ad8	Перехід, якщо біт дорівнює 1	$PC \leftarrow PC+3$, якщо $bit=1$, то $PC \leftarrow PC + ad8$	3	2
JCB bit, ad8	Перехід, якщо біт дорівнює 1 з наступним скиданням біта	$PC \leftarrow PC + 3$, якщо $bit = 1$, то $Bit \leftarrow 0$, $PC \leftarrow PC + ad8$	3	2
JNB bit, ad8	Перехід, якщо біт дорівнює 0	$PC \leftarrow PC + 3$, якщо $bit = 0$, то $PC \leftarrow PC + ad8$	3	2
DJNZ R, ad8	Декремент R і перехід, якщо не нуль	$PC \leftarrow PC + 2$, $R \leftarrow R - 1$, якщо $R \neq 0$, то $PC \leftarrow PC + ad8$	2	2
JNZ ad, ad8	Декремент байта з прямою адресацією і перехід, якщо не нуль	$PC \leftarrow PC + 2$, $ad \leftarrow ad - 1$, якщо $ad \neq 0$, то $PC \leftarrow PC + ad8$	3	2
CJNE A, ad, ad8	Порівняння A байта з прямою адресацією і перехід, якщо не дорівнює	$PC \leftarrow PC + 3$, якщо $A \neq ad$, то $PC \leftarrow PC + ad8$ Якщо $A < ad$, то $C = 1$, інакше $C = 0$	3	2

Продовж. табл. 6.11

Мнемокод	Опис	Алгоритм	Байт	Цикл
CJNE A, #data, ad8	Порівняння *A* з константою і перехід, якщо не дорівнює	$PC \leftarrow PC + 3$, якщо $A \leftarrow$ *#data*, то $PC \leftarrow PC +$ *ad8* Якщо $A <$ *#data*, то $C = 1$, інакше $C = 0$	3	2
CJNE R, #data, ad8	Порівняння регістра з константою і перехід, якщо не дорівнює	$PC \leftarrow PC + 3$, якщо $R \neq$ *#data*, то $PC \leftarrow PC +$ *ad8* Якщо $R <$ *#data*, то $C = 1$, інакше $C = 0$	3	2
CJNE @r, #data, ad8	Порівняння байта в РПД з константою і перехід, якщо не дорівнює	$PC \leftarrow PC + 3$, якщо $M_{РПД}(r) \neq$ *#data*, то $PC \leftarrow PC +$ *ad8* Якщо $M_{РПД}(r) <$ *<#data*, то $C = 1$, інакше $C = 0$	3	2
LCALL ad16	Довгий виклик підпрограми	$PC \leftarrow PC + 3$, $SP \leftarrow SP + 1$, $M_{РПД}(SP) \leftarrow PC0$–7, $SP \leftarrow SP + 1$, $M_{РПД}(SP) \leftarrow PC8$–15, $PC \leftarrow ad16$	3	2
ACALL ad11	Абсолютний виклик підпрограми всередині сторінки 2 Кбайт	$PC \leftarrow PC + 2$, $SP \leftarrow SP + 1$, $M_{РПД}(SP) \leftarrow PC_{0-7}$, $SP \leftarrow SP + 1$, $M_{РПД}(SP) \leftarrow PC_{8-15}$, $PC_{0-10} \leftarrow ad11$	2	2
RET	Повернення з підпрограми*	$PC_{8-15} \leftarrow M_{РПД}(SP)$, $SP \leftarrow SP - 1$, $PC_{0-7} \leftarrow M_{РПД}(SP)$, $SP \leftarrow SP - 1$	1	2
RETI	Повернення з підпрограми обробки переривання**	$PC_{8-15} \leftarrow M_{РПД}(SP)$, $SP \leftarrow SP - 1$, $PC_{0-7} \leftarrow M_{РПД}(SP)$, $SP \leftarrow SP - 1$	1	2
NOP	Немає операції	$PC \leftarrow PC + 1$	1	1

Примітки: * Команда не впливає на прапорці.

** Команда встановлює логіку переривань: дозволяє приймання переривань з рівнем пріоритету, що дорівнює рівню переривання, яке перед цим оброблялося.

Таблиця 6.12. **Вплив команд на прапорці**

Операція	Команда	Прапорці		
		OV	*C*	*AC*
Пересилання	*MOV PSW, source*	+	+	+
Додавання	*ADD, ADDC, SUBB*	+	+	+
Множення	*MUL*	+	0	–
Ділення	*DIV*	+	0	–
Порівняння і перехід	*CJNE*	–	+	–
Десяткова корекція	*DAA*	–	+	+
Зсув	*RRC, RLC*	–	+	–
Керування прапорцями	*SETB C*	–	1	–
	CLR C	–	0	–
	CPL C	–	$\bar{C}$	–
Команди роботи з бітами	*ANL C, bit*	–	+	–
	ANL C,/bit	–	+	–
	ORL C, bit	–	+	–
	ORL C,/bit	–	+	–
	MOV C, bit	–	+	–

Примітка. Прапорець парності *P* встановлюється за кожною командою, що модифікує вміст акумулятора згідно із цим вмістом. Прапорець нуля *Z* фізично не існує, однак під час виконання команд *JZ*, *JNZ* відбувається порівняння *A* з нулем.

6.3. Розширення можливостей однокристальних мікроконтролерів

Для розширення можливостей ОМК використовують: розширення пам'яті програм; розширення пам'яті даних; розширення простору введення-виведення.

Розширення пам'яті програм. Розширення пам'яті програм до 64 Кбайт виконують під'єднанням зовнішніх ВІС ПЗП до ОМК. При цьому якщо на вивід $\overline{EA}$ подано логічну одиницю, то використовується як внутрішня, так і зовнішня пам'ять програм: якщо адреса $00-0FFFH$, звернення відбуваються до внутрішньої, а якщо адреса $1000H-FFFFH$, – до зовнішньої пам'яті програм. У разі подання на вивід $\overline{EA}$ логічного нуля звернення відбуваються тільки до зовнішньої пам'яті програм.

Функціональну схему з'єднання зовнішньої пам'яті програм наведено на рис. 6.22. Через порти *P*0 і *P*2 передаються молодша і старша частини 16-розрядної адреси комірки зовнішньої пам'яті, що супроводжується сигналом *ALE*. Адреса утримується на виводах порту *P*2 впродовж усього машинного циклу звернення до пам'яті, а на виводах порту *P*0 — лише впродовж одного стану машинного циклу. Тому молодша частина адреси запам'ятовується у регістрі-замку *RG* за сигналом *ALE*. Після видавання молодшого байта адреси порт *P*0 переходить у високоімпедансний стан та очікується надходження даних із ПЗП.

Порт *P*0 функціонує як мультиплексована шина адреси-даних. Сигнал $\overline{PME}$ формується двічі за машинний цикл. Цей сигнал дозволяє вибірку байта з ПЗП. Тому впродовж одного машинного циклу вибирається два байти команди. Якщо команда однобайтова, другий байт ігнорується. Цей байт обирається під час переходу до наступної команди.

На рис. 6.23 зображено часові діаграми машинних циклів читання зовнішньої пам'яті програм у таких випадках: 1) команда звернення до зовнішньої пам'яті даних *MOVX* не виконується (рис. 6.23, *а*); 2) команда *MOVX* виконується (рис. 6.23, *б*). На рис. 6.23 позначено:

PCL out — видавання молодшого байта лічильника команд *PC*;

PCH out — видавання старшого байта лічильника команд *PC*;

DPH — видавання старшого байта регістра *DPTR*;

*P*2 *out* — видавання вмісту регістра фіксатора порту *P*2;

Addr out — видавання молодшого байта адреси зовнішньої пам'яті даних з регістрів *R*0, *R*1 або регістра *DPL*;

D in — введення байта команди або даних.

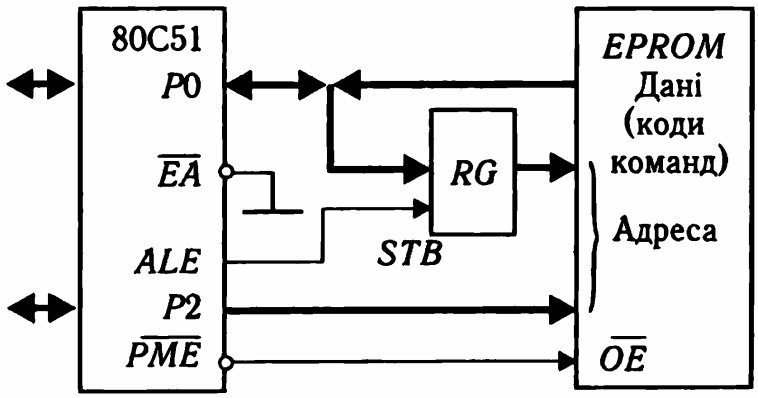
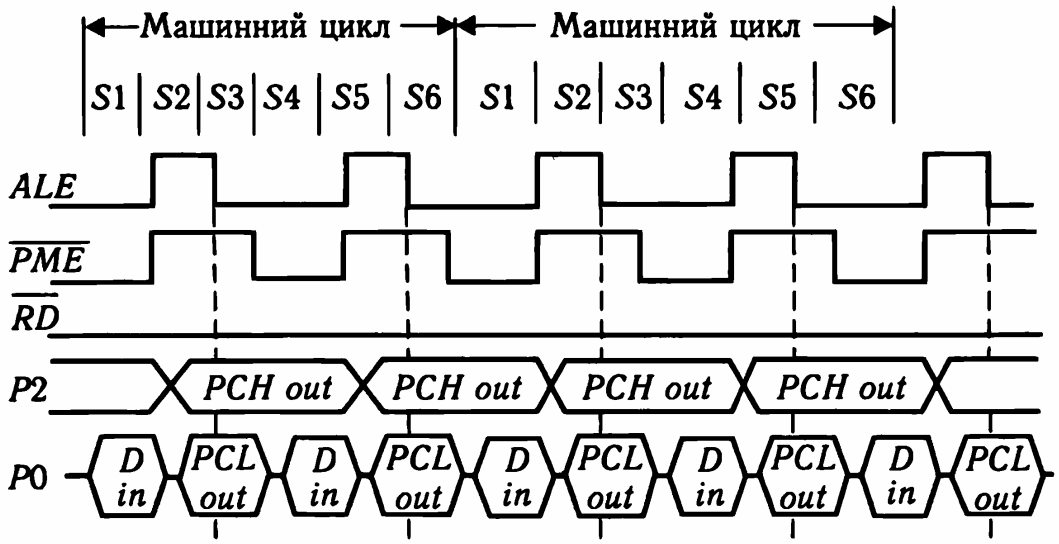
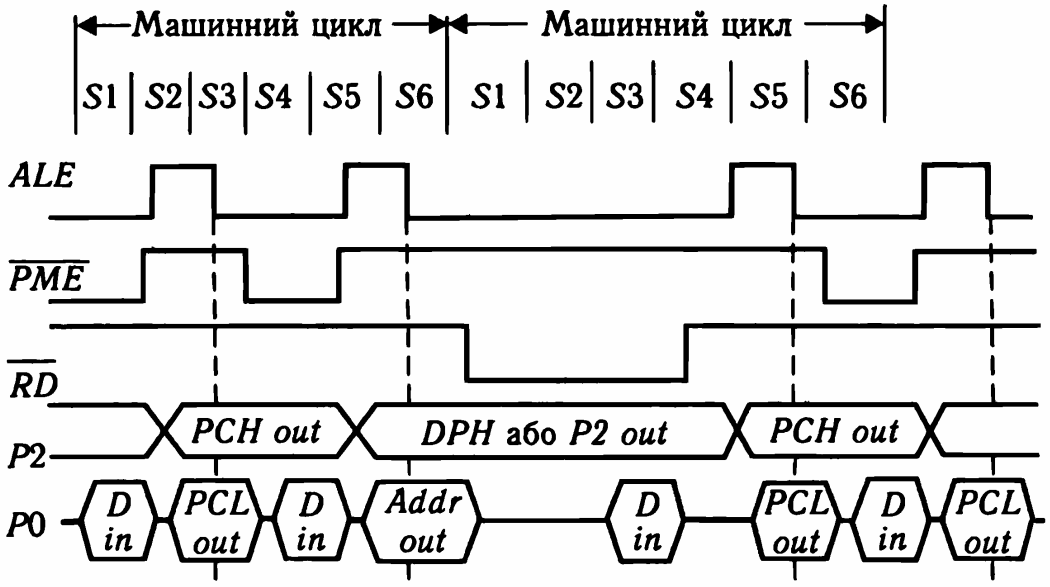


Рис. 6.22. Функціональна схема з'єднання зовнішньої пам'яті програм з ОМК



а



б

Рис. 6.23. Часові діаграми машинних циклів читання зовнішньої пам'яті програм:
а — команда *MOVX* не виконується; *б* — команда *MOVX* виконується

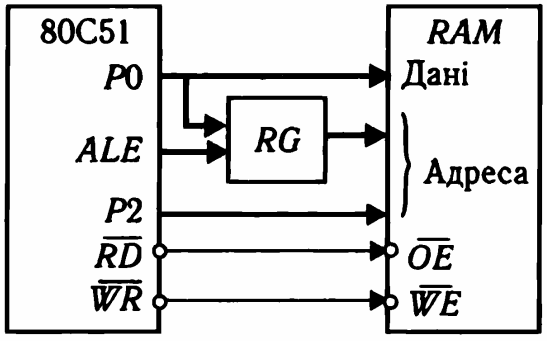
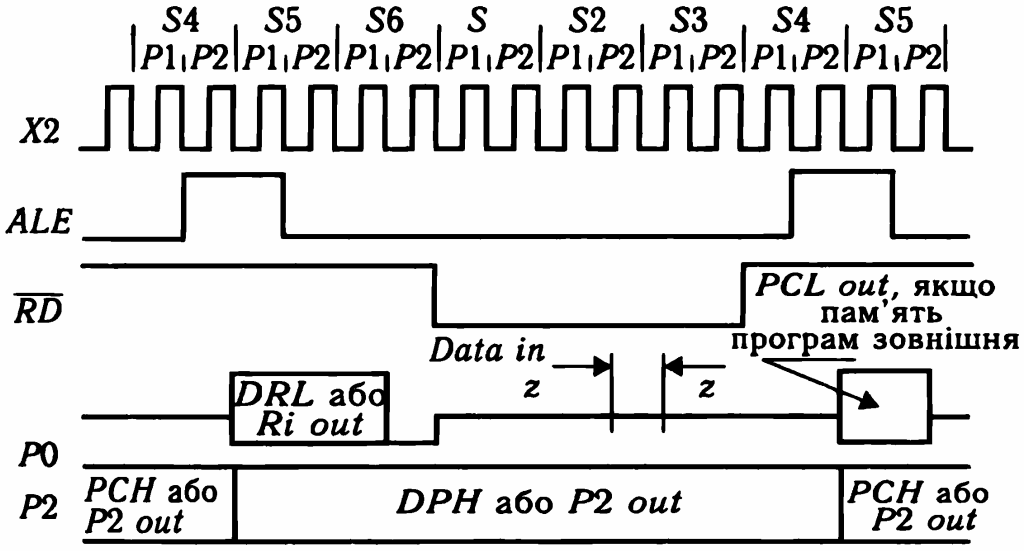


Рис. 6.24. Функціональна схема з'єднання зовнішньої пам'яті даних з ОМК

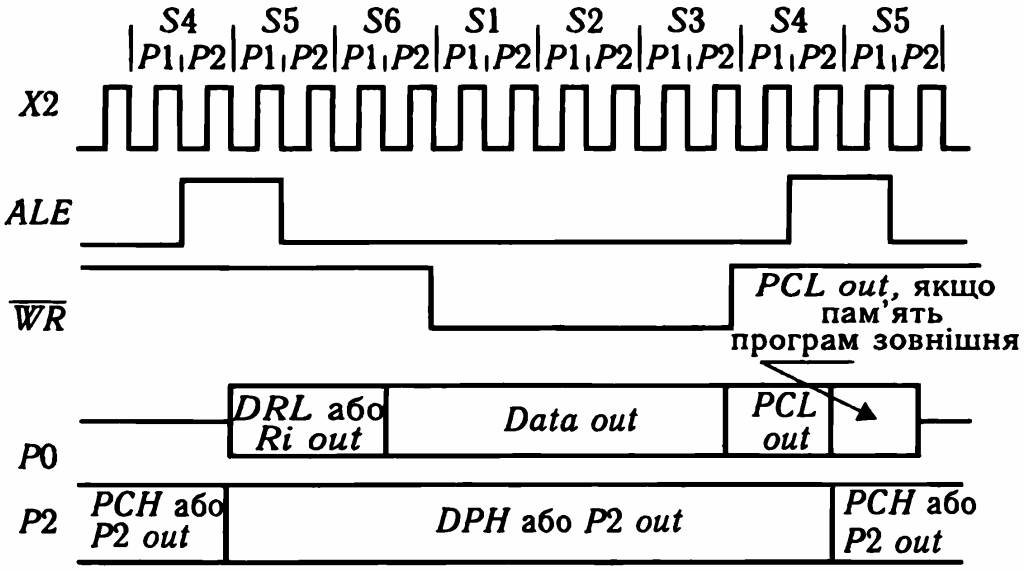
Як видно з рис. 6.23, *б* під час виконання команди *MOVX* сигнал *ALE* формується у другому машинному циклі лише один раз.

Якщо ОМК працює з внутрішньою пам'яттю програм, сигнал $\overline{PME}$ не формується й адреса на порти *P*0 і *P*2 не видається. Однак сигнал *ALE* і в цьому випадку формується двічі за період, якщо не виконується команда *MOVX*. Вивід *ALE* можна використовувати як вихідний сигнал синхронізації.

Розширення пам'яті даних. В ОМК передбачено можливість розширення пам'яті даних до 64 Кбайт з'єднан-



а



б

Рис. 6.25. Часові діаграми циклів: *а* — читання зовнішньої пам'яті даних; *б* — запису у зовнішню пам'ять даних

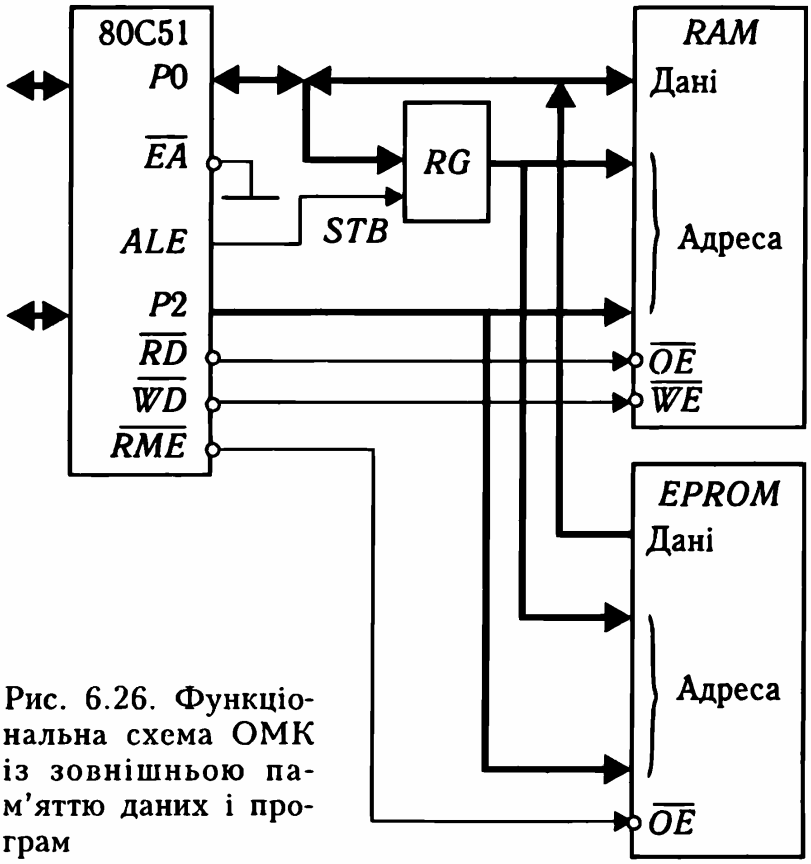


Рис. 6.26. Функціональна схема ОМК із зовнішньою пам'яттю даних і програм

ням зовнішніх ВІС пам'яті з ОМК. На рис. 6.24 зображено функціональну схему з'єднання зовнішньої пам'яті даних з ОМК.

Часові діаграми циклів читання і запису в зовнішню пам'ять даних зображено на рис. 6.25, *а, б*. З'єднання пам'яті даних з ОМК аналогічне з'єднанню пам'яті програм.

За потреби в ОМК можуть одночасно використовуватися зовнішня пам'ять даних та пам'ять програм (рис. 6.26).

Розширення простору введення-виведення. Одну з можливих схем з'єднання контролера клавіатури і дисплея з ОМК зображено на рис. 6.27.

За такого з'єднання адреса контролера входить до адресного простору зовнішньої пам'яті даних. Залежно від типу звернення (керування або передавання даних) лінія *P*1.0 порту *P*1 з'єднується з лінією *A*0 контролера і відбувається встановлення або скидання перед зверненням ОМК до контролера. Вхід вибірки кристала контролера з'єднаний із загальною точкою, тому контролер завжди готовий до обміну інформацією з ОМК. Вихід сигналу запиту переривання *IRQ* контролера з'єднаний з лінією *P*1.1 ОМК. Опитування стану виходу *IRQ* здійснюється для визначення факту натискання клавіші.

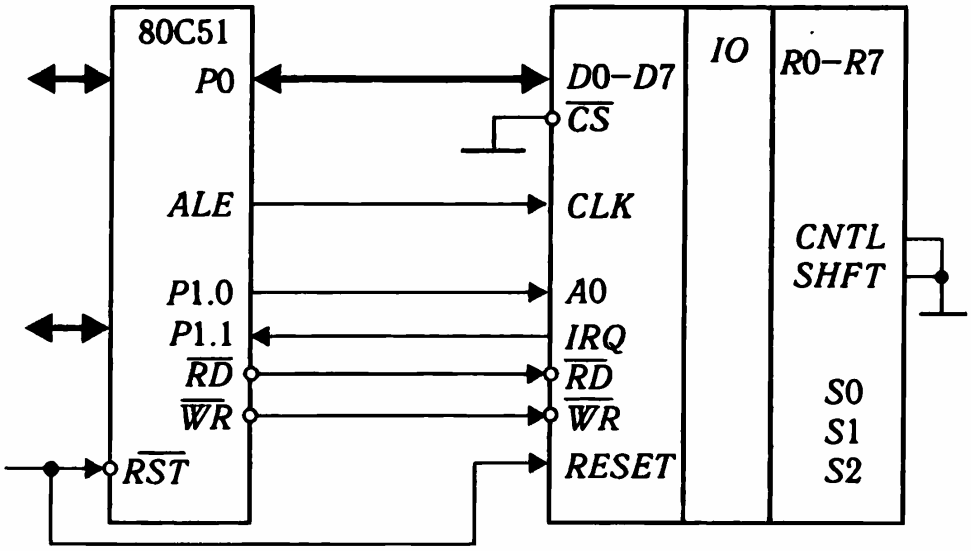


Рис. 6.27. З'єднання контролера клавіатури і дисплея з ОМК

Шина даних і лінії читання-запису контролера об'єднуються безпосередньо з відповідними лініями ОМК. На вхід *CLK* подається сигнал частотою приблизно 2 МГц з виходу *ALE*.

6.4. Застосування однокристального мікроконтролера 83C51FA для керування двигуном постійного струму

Двигуни постійного струму широко використовують у промислових і побутових пристроях. У багатьох випадках важливим є прецизійне керування швидкістю обертання двигуна та можливість зміни напряму обертання. Наприклад, двигун постійного струму у побутовій техніці магнітного звукозапису має обертатися зі сталою швидкістю. Зміна напряму обертання досягається зміною полярності напруги, що подається або на колекторну обмотку або на обмотку збудження.

Використання ОМК 83C51FA дає змогу здійснювати керування двигуном постійного струму з реалізацією широтно-імпульсної модуляції напруги, яка подається на обмотку збудження чи на колекторну обмотку. На базі ОМК 83C51FA можна запрограмувати до п'яти широтно-імпульсних модуляторів.

Однокристальний мікроконтролер 83C51FA — це 8-розрядний мікроконтролер, основою якого є архітектура 8051, але він має кілька нових функцій, зокрема існує *масив про-*

грамних лічильників — PCA (Programmable Counter Array). Масив складається з 16-розрядного таймера *PCA* та п'яти окремих модулів. Таймер *PCA* має два 8-розрядних регістри — *CL* (молодший байт) та *CH* (старший байт), які доступні для всіх модулів. Причому таймер *PCA* можна запрограмувати на використання вхідних даних з чотирьох різних джерел. Максимальна частота лічення — 4 МГц, тобто 1/4 частоти генератора. Окремі виводи порту *P*1 використовуються для взаємодії кожного модуля і таймера із зовнішніми пристроями. Якщо виводи порту не використовуються для роботи модулів *PCA*, їх можна використати як лінії звичайного порту введення-виведення.

Модулі *PCA* можуть бути запрограмовані на режим захоплення події (capture) або режим порівняння, який має такі напрями: програмний таймер; режим високошвидкісного виведення, режим широтно-імпульсного модулятора (*PWM*); режим сторожового таймера (watchdog-таймера) (лише для четвертого модуля). Кожний модуль має 8-розрядний регістр режиму — *CCAPMn* (табл. 6.13) та 16-розрядний регістр порівняння-захоплення, який складається з двох 8-розряд-

Таблиця 6.13. **Позначення бітів регістра режиму *CCAPMn***

Біт	Позначення	Призначення
7	—	—
6	*ECOMn*	Біт дозволу функції порівняння; *ECOMn* = 1 для функцій, що потребують порівняння вмісту регістрів порівняння-захоплення із вмістом 16-розрядного таймера, наприклад функцій програмованого таймера, високошвидкісного виведення, watchdog-таймера
5	*CAPPn*	Біт захоплення за переднім фронтом сигналу
4	*CAPNn*	Біт захоплення за заднім фронтом сигналу
3	*MATn*	Біт визначення збігу вмісту регістрів захоплення-порівняння із вмістом 16-розрядного таймера
2	*TOGn*	Біт дозволу виведення за умовою збігу вмісту регістрів захоплення-порівняння із вмістом 16-розрядного таймера
1	*PWMn*	Біт дозволу генерації сигналу широтно-імпульсного модулятора за збігом молодшого байта вмісту регістра порівняння-захоплення та молодшого байта вмісту таймера *PCA*
0	*ECCCFn*	Біт дозволу генерації переривання за прапорцем порівняння-захоплення *CCFn* регістра *CCON*

них регістрів *CCAPnL* і *CCAPnH*, де *n* може набувати значення від 0 до 4.

Установленням відповідних бітів у регістрах режимів *CCAPMn* можна запрограмувати кожний модуль на функціонування в одному з режимів — захоплення події або порівняння.

Режим захоплення події. *Подією* називають будь-яку зміну рівня сигналу на входах. Однокристальний мікроконтролер можна запрограмувати на визначення таких подій:

- кожного переходу рівня вхідного сигналу зі стану «0» у стан «1» (*CAPPn* = 1);
- кожного переходу рівня вхідного сигналу зі стану «1» у стан «0» (*CAPNn* = 1);
- будь-якого переходу рівня сигналу (*CAPPn* = 1 та *CAPNn* = = 1).

Після того, як запрограмована подія відбулася, час настання цієї події, відлічений таймером PCA разом з інформацією про стан входів записується у стек подій *FIFO*. Операцію запису часу події у стек називають *захопленням події*. Якщо встановлено біт дозволу *ECCFn*, захоплення події супроводжується генерацією запиту переривання ЦП на обслуговування модуля.

Режим порівняння. Якщо модуль запрограмовано на функціонування в одному з підрежимів порівняння (програмований таймер, високошвидкісний вивід, *watchdog*-таймер), програма завантажує у регістри захоплення-порівняння величину, яка порівнюється з вмістом 16-розрядного таймера; ОМК генерує переривання у разі збігу цих величин.

У підрежимі *програмованого таймера* встановлюється прапорець переповнення у разі збігу значень таймера і регістра захоплення-порівняння.

Підрежим *високошвидкісного виводу* призначений для генерації заданих подій у певний час. Час задається значенням регістру захоплення-порівняння *CCAPn*.

У підрежимі *сторожового таймера* генерується переривання, якщо деяка ділянка програми виконується за більший час, ніж заданий у регістрі. Це дає змогу уникнути «зависання» програми.

Підрежим широтно-імпульсного модулятора — єдиний режим, що використовує лише 8-розрядний регістр захоплення-порівняння. У старший байт (*CCAPnH*) обраного модуля завантажується значення від 0 до *FFH*. Це значення переноситься у молодший байт того самого модуля і порівнюється з молодшим байтом регістра таймера *PCA*. За умови *CL* < *CCAPnL* на відповідному виводі встановлюється стан

логічного нуля; за умови *CL* > *CCAPnL* — стан логічної одиниці.

Крім регістрів *CCAPMn* для забезпечення роботи таймера *PCA* введено ще два регістри спеціальних функцій — *CCON* та *CMOD*. Регістр *CCON* (табл. 6.14) допускає бітову адресацію. Адреса регістра *CCON* — 0*D*8*H*, значення для скидання — 00x00000*B*.

Регістр *CMOD* (табл. 6.15) не допускає адресацію окремих бітів. Адреса регістра *CMOD* — 0*D*9*H*, значення для скидання — 00xxx000*B*.

Якщо один з модулів запрограмований у режим широтно-імпульсного модулятора, у старший байт регістра порівняння слід завантажити значення від 0 до 255, яке визначає коефіцієнт заповнення широтно-імпульсного модулятора. Для ОМК 83C51*FA* завантаження 0 в *CCAPnH* відповідає коефі-

Таблиця 6.14. **Позначення бітів регістра *CCON***

Біт	Позначення	Призначення
7	*CF*	Прапорець переповнення таймера
6	*CR*	Запуск таймера PCA
5	—	Не використовується
4	*CCF*4	Прапорці модулів, які використовуються для визначення модуля, що генерує переривання *PCA*
3	*CCF*3	
2	*CCF*2	
1	*CCF*1	
0	*CCF*0	

Таблиця 6.15. **Позначення бітів регістра *CMOD***

Біт	Позначення	Призначення
5	—	Не використовується
4	—	–«–
3	—	–«–
2	*CPS*1	Біти, що визначають джерело тактування таймера *PCA*: 00 — внутрішній генератор, *Fosc*/12 01 — внутрішній генератор, *Fosc*/4 10 — переповнення таймера 0 11 — зовнішній сигнал (введення з *P*1.2)
1	*CPS*0	
0	*ECF*	Дозвіл переривання за прапорцем *CF*

Таблиця 6.16. **Часові діаграми і значення коефіцієнта заповнення широтно-імпульсного модулятора**

Коефіцієнт заповнення	Значення регістра *CCAPnH*	Вихідний сигнал широтно-імпульсного модулятора
1	00	
0,9	25	
0,5	128	
0,1	230	
0,004	255	

цієнту заповнення 1, а 255 (*OFFh*) — коефіцієнту заповнення 0,004. Часові діаграми та значення коефіцієнта заповнення широтно-імпульсного модулятора і регістра *CCAPnH* зображено в табл. 6.16.

Запуск таймера *PCA* здійснюється встановленням біта *CR* (див. табл. 6.14) регістра *CCON*, який допускає бітову адресацію. Встановлення і скидання цього біта здійснюється відповідними бітовими командами — *CLR bit* на *SETB bit* (див. табл. 6.11).

Приклад 6.1. Запрограмувати модуль 2 для генерації широтно-імпульсного модулятора-сигналу для ланцюга керування двигуном.

Програма має вигляд:

```
MOV     CMOD,#06        ; Обирається зовнішній тактовий сигнал
MOV     CCAPM2,#42H     ; Встановлюється режим
                        ; широтно-імпульсного модулятора
MOV     CCAP2H,#0       ; Значення нуля відповідає коефіцієнту
                        ; заповнення 100 % (5 В)
SETB    CR              ; Запуск таймера
END
```

Принципову схему модуля керування двигуном постійного струму зображено на рис. 6.28 .

Схема містить ОМК 83C51FA з колами скидання і синхронізації, спеціалізовану ВІС драйвера L293, двигун та імпульсний датчик швидкості 30137. Імпульси ТТЛ-рівня з виходу датчика надходять на вхід *P*1.6 ОМК. Залежно від частоти цих імпульсів ОМК збільшує або зменшує коефіцієнт за-

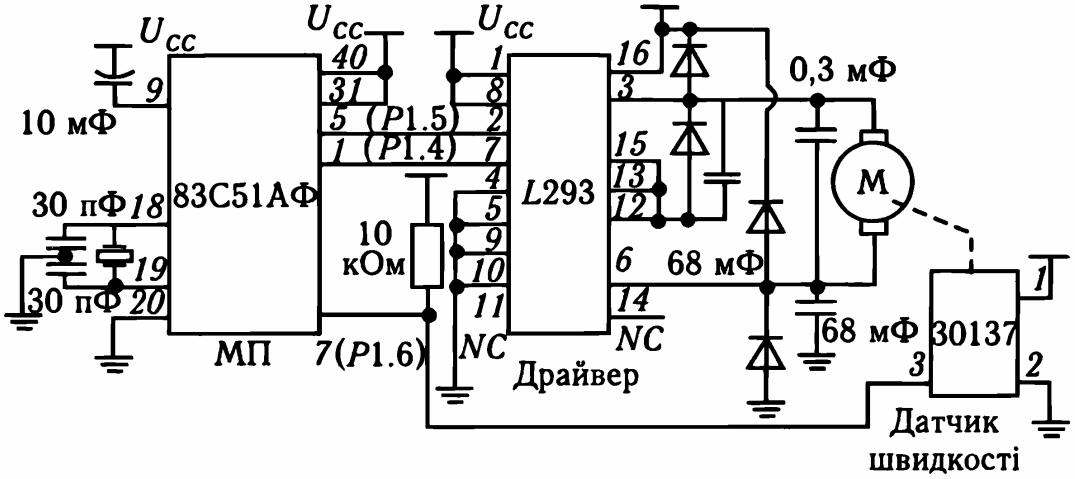


Рис. 6.28. Принципова схема модуля керування двигуном постійного струму

повнення широтно-імпульсного модулятора на виводах $P1.4$, $P1.5$, реалізуючи таким чином регулювання швидкості обертання двигуна.

6.5. Архітектура і функціональні можливості 16-розрядних однокристальних мікропроцесорів серії MCS 196/296

Загальна характеристика. 16-Розрядні ОМК MCS 196/296 виробництва компанії Intel використовуються у вмонтованих МПС керування в авіаційній та автомобільній промисловості, верстато- і роботобудуванні, енергетиці та електромеханіці, пристроях побутової техніки. Нині серія 16-розрядних ОМК містить понад 30 типів контролерів продуктивністю від 1 до 16 млн операцій типу «регістр-регістр» за секунду, оскільки операції типу «регістр-регістр» є найбільш швидкодіючими й оперують лише з внутрішніми регістрами МП. Прикладом таких операцій є *MOV R*0, *R*1; *ADD A*, *R*3. Основні характеристики ВІС наведено у табл. 6.17.

Відповідно до типу інтегрованих на кристал периферійних пристроїв розрізняють такі класи ОМК серії 196/296:

- ОМК з інтегрованими пристроями високошвидкісного введення-виведення;
- ОМК з інтегрованими процесорами подій;
- ОМК з інтегрованими засобами керування двигунами;
- ОМК з інтегрованими засобами цифрового оброблення.

Однокристальні мікроконтролери з інтегрованими пристроями високошвидкісного введення-виведення. 16-Роз-

Таблиця 6.17. **Основні технічні характеристики 16-розрядних ОМК**

Тип BIC	Технічні характеристики											
	Тактова частота, МГц	Адресний простір, Кбайт	Ємність ПЗП, Кбайт	Ємність регістрового ОЗП, байт	Ємність флеш-пам'яті програм, байт	Кількість таймерів	Кількість каналів АЦП	Кількість ліній введення-виведення	Кількість послідовних портів	Режим тестування	Тип процесора подій	Характерні особливості
	Мікроконтролери з вбудованим пристроєм високошвидкісного введення-виведення											
8 × C196KB	12,16	64	8	232	–	2	8	48	1	+	Модуль високошвидкісного введення-виведення (4 входи, 6 виходів)	–
8 × C198	16	64	8	232	–	2	0 або 4	48	1	+	Модуль високошвидкісного введення-виведення (4 входи, 6 виходів)	8-Розрядна зовнішня шина, менша вартість
8 × C196KC	16,2	64	16	488	–	2	8	48	1	+	Модуль високошвидкісного введення-виведення (4 входи, 6 виходів)	ШІМ-Генератор, сервер периферійних транзакцій
8 × C196KD	16,2	64	32	1000	–	2	8	48	1	+	Модуль високошвидкісного введення-виведення (4 входи, 6 виходів)	ШІМ-Генератор, сервер периферійних транзакцій

	Мікроконтролери з вмонтованими процесорами подій											
8 × C196KR	16	64	16	488	256	2	8	56	2	+	10 каналів захоплення-порівняння, 2 канали лише порівняння	Блок сегментації пам'яті, порт для міжпроцесорних комунікацій
8 × C196KT	16	64	32	1000	512	2	8	56	2	+	10 каналів захоплення-порівняння, 2 канали лише порівняння	Контролер шини з розширеними можливостями, порт для міжпроцесорних комунікацій
8 × C196NP	25	100	4	1000	–	2	–	33	1	+	4 канали процесора подій	Низький рівень живлення 3,3 В, триканальний ШІМ-генсратор, сервер периферійних транзакцій, 4 зовнішніх переривання
8 × C196NT	16	100	32	1000	512	2	4	56	2	+	10 каналів захоплення-порівняння, 2 канали лише порівняння	–
8 × C196NU	50	100	–	1000	–	2	–	56	1	+	4 канали процесора подій	32-Розрядний акумулятор
	Мікроконтролери для керування двигунами											
8 × C196MC	16	64	16	488	–	2	13	53	1	+	4 канали захоплення-порівняння, 4 лише порівняння	Триканальний ШІМ-генератор, процесор подій процессор транзакцій, триканальний генератор періодичних сигналів

Продовж. табл. 6.17

Тип BIC	Технічні характеристики											
	Тактова частота, МГц	Адресний простір, Кбайт	Ємність ПЗП, Кбайт	Ємність регістрового ОЗП, байт	Ємність флеш-пам'яті програм, байт	Кількість таймерів	Кількість каналів АЦП	Кількість ліній введення-виведення	Кількість послідовних портів	Режим тестування	Тип процесора подій	Характерні особливості
8×C196MD	16	64	16	488	–	2	14	64	1	+	6 каналів захоплення-порівняння, 6 каналів лише порівняння	Вмонтований генератор
8×C196MH	16	64	32	744	–	2	8	52	2	+	2 канали захоплення-порівняння, 4 канали лише порівняння	–
	Мікроконтролери з інтегрованими засобами цифрового оброблення											
8×0C296SA	50	600	32	512	2K	2		64	1		4 канали захоплення-порівняння	Збільшення в 2 або 4 рази тактової частоти, 19 джерел переривань, із них 4 зовнішніх, трикануальний ШІМ-генератор, набір процесорів подій, вмонтований модуль вибірки зовнішніх пристроїв

рядні ОМК з інтегрованими пристроями високошвидкісного введення-виведення даних (*HSIO — High-Speed Input/Output*) орієнтовані на вирішення завдань керування об'єктами і процесами у реальному масштабі часу. ОМК широко використовують не лише в комп'ютерній техніці (принтери, плотери), а й в електромеханіці, робототехніці, військовій техніці завдяки наявності інтегрованого АЦП, модуля *HSIO*, процесора периферійних трансакцій та порту послідовного зв'язку. Конкретні ОМК вирізняються тактовою частотою, кількістю послідовних портів, входів АЦП, ємністю інтегрованої на кристал пам'яті, наявністю тих або інших периферійних блоків, наприклад генератора періодичних сигналів.

Структурна схема ОМК (рис. 6.29) містить: модуль ЦП; контролер пам'яті (КП); контролер переривань (*IC — Interrupt Controller)*; сервер периферійних трансакцій (*PTS — Peripheral Transaction Server*); ПЗП; блок АЦП;

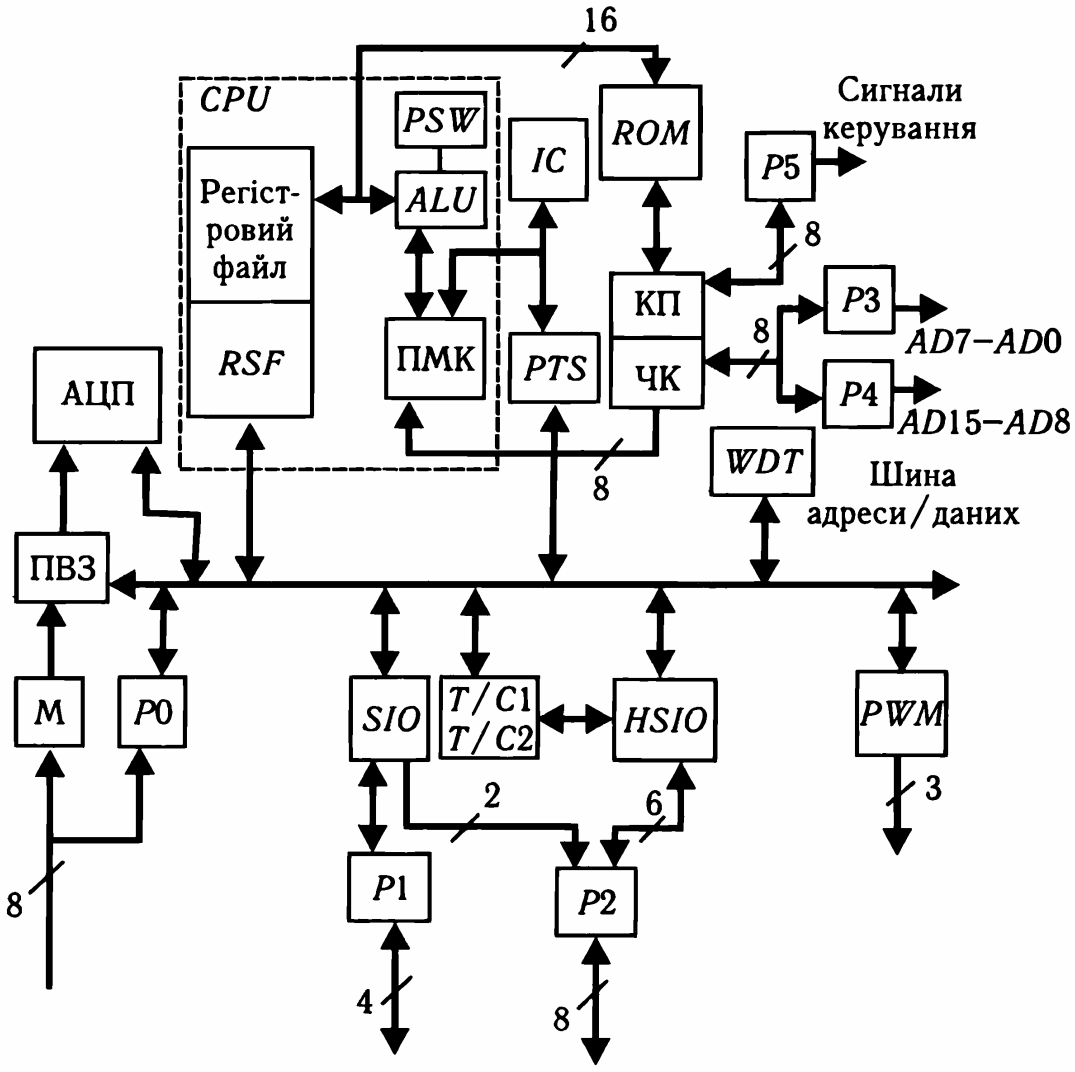


Рис. 6.29. Структурна схема ОМК з інтегрованим пристроєм високошвидкісного введення-виведення

вбудований триканальний ШІМ-генератор (*PWM*); послідовний порт (*SIO*); порти введення-виведення (*P*0—*P*5); *watchdog*-таймер (*WDT*).

Модуль ЦП складається з регістрового АЛП, регістра слова стану процесора *PSW*, пристрою мікропрограмного керування (ПМК) і регістрового файла. *Регістровий файл* — масив регістрів, тобто ОЗП статичного типу. Він складається з нижнього і верхнього регістрових файлів. Молодші 24 байти нижнього регістрового файла належать до регістрів спеціальних функцій. У нижньому регістровому файлі розміщено також покажчик стеку. Решта комірок регістрового файла, включаючи комірки верхнього регістрового файла, є оперативною пам'яттю загального призначення.

Центральний процесор пов'язаний з контролером пам'яті, контролером переривань і вмонтованими периферійними пристроями за допомогою внутрішньої 16-розрядної шини. Крім того, існує додаткова 8-розрядна шина для безпосереднього передавання байтів команди з контролера пам'яті у регістр команд, який знаходиться у пристрої мікропрограмного керування.

Контролер пам'яті здійснює операції вибирання-записування даних у зовнішню або внутрішню пам'ять даних або команд. До складу контролера пам'яті входить чотирибайтова черга команд (ЧК). Функціонування контролера пам'яті аналогічне функціонуванню шинного інтерфейсу МП 8086 (див. п. 3.2). Звернення ЦП відбуваються лише до регістрового файла.

Контролер переривань призначений для ефективного керування у реальному часі. Обробка запитів переривань здійснюється з урахуванням установленого рівня пріоритету.

Як запити переривань можуть використовуватися:

- зовнішні сигнали *NMI* або *EXTINT*;
- запити від інтегрованих периферійних пристроїв;
- команди викликів підпрограм обробки переривань.

Сервер периферійних трансакцій виконує функції периферійного процесора введення-виведення і звільняє ЦП від виконання деяких типових операцій введення-виведення. *Трансакцією* називають виконання цілісної операції над даними або пакетами даних, наприклад, введення або виведення пакета даних відповідно до заданого протоколу у послідовному каналу зв'язку, пересилання даних з однієї області пам'яті в іншу, сканування АЦП, тобто зчитування результатів, запис у пам'ять та опитування каналу АЦП. Сервер периферійних трансакцій працює виключно за перериванням (*PTS*-переривання), але обслуговує їх не програмно, а апаратно на

мікропрограмному рівні. При цьому підвищується швидкодія обчислень, і деякі *PTS*-переривання можуть оброблятися за час виконання однієї звичайної команди.

Перед тим, як дозволити *PTS*-переривання, слід задати: бажаний режим роботи, кількість циклів передавання даних, адреси джерел і приймачів даних. Це здійснюється програмно записуванням необхідної інформації в керуючий блок сервера периферійних трансакцій, що знаходиться у регістровому файлі. Після закінчення програми обслуговування *PTS*-переривання генерується звичайне переривання для переініціалізації сервера на нову задачу.

Постійний запам'ятовувальний пристрій. Залежно від типу BIC як внутрішня пам'ять програм використовується програмований маскою або однократно програмований ПЗП ємністю від 8К × 8 до 32К × 8 байт.

Блок АЦП призначений для введення аналогових сигналів і складається з аналогового мультиплексора *M* (наприклад, на 8 каналів для ОМК 8 × C196KD), пристрою вибірки-зберігання (ПВЗ) і 8- або 10-розрядного АЦП. За допомогою регістра спеціальних функцій *AD_TIME* задається тривалість вибирання і перетворення, за допомогою регістра *AD_COMMAND* — номер вхідного каналу, для якого треба виконати перетворення, та умови запуску процесу перетворення (8 або 10 розрядів АЦП). Якщо використати 10-розрядний АЦП, то результат зчитується з регістра *AD_RESULT* або за опитуванням, або за перериванням (за готовністю АЦП). Тривалість вибірки 10-розрядного АЦП становить 1 мкс, тривалість перетворення у 10-розрядному режимі — 10—20, у 8-розрядному режимі — 7—20 мкс.

Модуль високошвидкісного введення-виведення призначений для формування періодичних у часі сигналів (наприклад, сигналу ШІМ для прямого цифрового керування інверторами напруги приводів змінного чи постійного струму) та безпосереднього вимірювання часових інтервалів між зовнішніми подіями без додаткових периферійних пристроїв (наприклад, вимірювання частоти, шпаруватості, періоду, фазового зсуву). *Подією* вважається будь-яка зміна потенціалу зовнішнього сигналу на входах *HSI*.0—*HSI*.3 ОМК.

До складу модуля входять: два таймери (*Timer*1 і *Timer*2) для точного відліку часових інтервалів; модуль високошвидкісного введення даних *HSI*; модуль високошвидкісного виведення даних *HSO*.

Модуль високошвидкісного введення апаратно контролює зміну потенціалу зовнішнього сигналу на кожному з чотирьох входів *HSI*.0—*HSI*.3 за допомогою вбудованої схе-

ми детектора перепадів. Ці зміни сприймаються ОМК як зовнішні події. Кожний з входів *HSI*.0—*HSI*.3 можна незалежно запрограмувати на визначення таких подій:

• кожного переходу значення вхідного сигналу зі стану логічної одиниці у стан логічного нуля;

• кожного переходу значення вхідного сигналу зі стану логічного нуля у стан логічної одиниці;

• кожного переходу значення вхідного сигналу зі стану логічної одиниці у стан логічного нуля та зі стану логічного нуля у стан логічної одиниці;

• восьми переходів значення вхідного сигналу зі стану логічної одиниці у стан логічного нуля.

Якщо запрограмована подія відбулася на одному або кількох входах, то час настання цієї події, відлічений таймером 1, разом з інформацією про стан усіх входів записується у *стек подій FIFO,* тобто відбувається *захоплення події.* Захоплення події супроводжується генерацією запиту переривання ЦП на обслуговування модуля *HSI* (*HSI Data Available Interrupt*). Якщо до захоплення події стек подій був порожнім, то запис здійснюється у *регістр зберігання* (*Holding Register*) інформації про подію, що відбулася, інакше запис здійснюється у стек. Модуль *HSI* може послідовно запам'ятовувати інформацію про максимум вісім подій: сім записів можуть зберігатися у стеку *FIFO* ємністю 7 × 20 біт та один запис — у регістрі зберігання.

Основна задача модуля високошвидкісного виведення полягає у генерації заданих подій у певний час. Існують внутрішні та зовнішні події. Внутрішні події, не пов'язані зі зміною потенціалу на виводах ОМК, використовують, наприклад, для запуску процесів:

• перетворення у вбудованому АЦП у певний момент часу;

• обслуговування програмно-реалізованих таймерів (*Software Timer*), які використовуються для відліку заданих досить великих часових інтервалів. При цьому одночасно може обслуговуватися до чотирьох таких таймерів.

Інформація про подію і тривалість генерації події у вигляді команди записується програмно в асоціативну пам'ять *CAM*-файл (*Content Addressable Memory*). Ємність *CAM*-файла дозволяє одночасно запам'ятовувати до 8 команд, формуючи таким чином деяку *програму генерації подій.* Кожна з цих команд задає:

• тип події (скидання або встановлення потенціалу на зовнішньому контакті, запуск АЦП, запуск програмованих таймерів);

• спосіб реагування ЦП на подію — інформація про те, чи буде генеруватися переривання для обслуговування події;

• номер таймера (1 чи 2), який використовують для контролю поточного реального часу;

• тривалість генерації події.

У кожному машинному такті (100 нс за тактової частоти 20 МГц) відбувається порівняння часу генерації події з реальним часом, що відлічує таймер. Якщо вони збігаються, то генерується задана командою подія; якщо подія відбулася, вона може бути вилучена зі списку подій або перезаписана знову.

Вбудований триканальний ШІМ-генератор. Модуль *HSIO* здатний реалізувати функцію ШІМ, але він має певні обмеження, пов'язані з необхідністю витрат часу звернення ЦП на його обслуговування. Це призводить до обмеження максимальної та мінімальної шпаруватості ШІМ-сигналів. Тому для генерації ШІМ-сигналів на високих частотах (до 40 кГц) краще використовувати вбудований триканальний ШІМ-генератор, який не має обмежень на скважність імпульсів. Частота ШІМ-генератора для всіх каналів є однаковою, а скважність регулюється від 1/255 до 1 з дискретністю 1/255. Недоліком ШІМ-генератора є обмежені можливості регулювання частоти. Так, для ВІС 8×С196KD за тактової частоти 20 МГц частота ШІМ-генератора може встановлюватися в одне з двох можливих значень — 39,1 або 19,5 кГц.

Вбудований ШІМ-генератор ефективно використовують для прямого цифрового керування електронними комутаторами, ключами інверторів напруги та струму, а також для формування аналогових сигналів задання у цифроаналогових системах керування.

Послідовний порт SIO — універсальний синхронно-асинхронний приймач-передавач, який може налагоджуватися на один з чотирьох режимів, один з яких є синхронним, а три — асинхронними. Швидкість обміну інформацією у послідовному порту програмно регулюється від кількох сотень бод (біт/с) до кількох десятків тисяч бод. Порт забезпечує роботу в режимі *Master-Slave*, що дає змогу увімкнути мікроконтролер у мультимікропроцесорну систему керування.

Однокристальний мікроконтролер з інтегрованими процесорами подій. 16-Розрядний ОМК з інтегрованим процесором подій, або *EPA*-мікроконтролер (*Event Processor Array* — масив процесорів подій), відрізняється від ОМК із вбудованим модулем *HSIO* наявністю потужного процесора подій, модифікованим блоком вбудованих таймерів-лічильників, більшою ємністю прямо адресованої зовнішньої пам'яті (від 1 до 16 Мбайт), розширеними інтерфейсними можливостями.

Модифікація блока вбудованих таймерів-лічильників полягає у тому, що обидва таймери реверсивні й можуть працювати як із внутрішнім, так і з зовнішнім тактуванням; є можливість каскадного вмикання, коли тактовими імпульсами таймера *T/C* 1 є імпульси переповнення від таймера *T/C* 0; обидва таймери допускають роботу у *квадратурному режимі*, коли дві послідовності прямокутних імпульсів, зсунутих на 90 електричних градусів, знімаються з датчика і вводяться безпосередньо у мікроконтролер. Цей режим зручний під час роботи з датчиками положення і/або швидкості, наприклад, з оптичними датчиками Хола тощо.

Масив процесора подій сумісний знизу з модулем високошвидкісного введення-виведення і призначений для розв'язання тих самих задач, але має деякі вдосконалення. Кожний канал процесора подій працює в одному з двох режимів — режимі захоплення зовнішніх подій або режимі генерації внутрішніх чи зовнішніх подій. Як базовий для кожного з каналів може використовуватися один з таймерів 1 або 2, який тактується за одним із таких способів:

- зовнішнє тактування за допомогою входу зовнішніх тактових імпульсів і входу напряму лічення;
- зовнішнє тактування двома послідовностями імпульсів у квадратурному режимі;
- внутрішнє тактування з можливістю зміни частоти вхідних тактових імпульсів через програмування коефіцієнта ділення;
- тактування сигналом переповнення іншого таймера.

Масив процесорів подій реагує на такі типи зовнішніх подій: перехід сигналу на зовнішньому виводі зі стану логічної одиниці у стан логічного нуля; перехід сигналу на зовнішньому виводі зі стану логічного нуля у стан логічної одиниці; перемикання рівня сигналу на зовнішньому виводі.

Кожний канал процесора подій генерує окремий запит на переривання, внаслідок чого значно спрощується та прискорюється обслуговування процесора подій з боку ЦП і не потребується ідентифікація джерела запиту.

Оскільки виконання функцій захоплення-порівняння кожним каналом здійснюється незалежно, часова дискретність визначення події або генерації сигналів збільшується в кілька разів. Так, за частоти 20 МГц часова дискретність *EPA*-мікроконтролера становить 200 нс, що вчетверо менше, ніж у *HSIO*-мікроконтролера з тією самою тактовою частотою.

Однокристальний мікроконтролер з інтегрованими засобами керування двигунами, або *MCA*-мікроконтролери

(*Motor Control Family*), призначені для прямого цифрового керування перетворювальними пристроями електроприводів. Структурну схему *MCA*-мікроконтролера серії 8С196МН зображено на рис. 6.30.

Крім розглянутих вище блоків ОМК (див. рис. 6.29), схема містить додаткові лінії портів та складніший генератор періодичних сигналів. Лінії семи портів ОМК мають таке функціональне призначення: порт 0 — функція АЦП; порт 1 — 4 лінії послідовного введення-виведення; порт 2 — дві лінії послідовного введення-виведення, два канали захоплення-порівняння, чотири канали порівняння процесора подій; порти 3,4 — суміщена шина адреси-даних; порт 5 — вироблення сигналів керування; порт 6 — функція генератора періодичних сигналів.

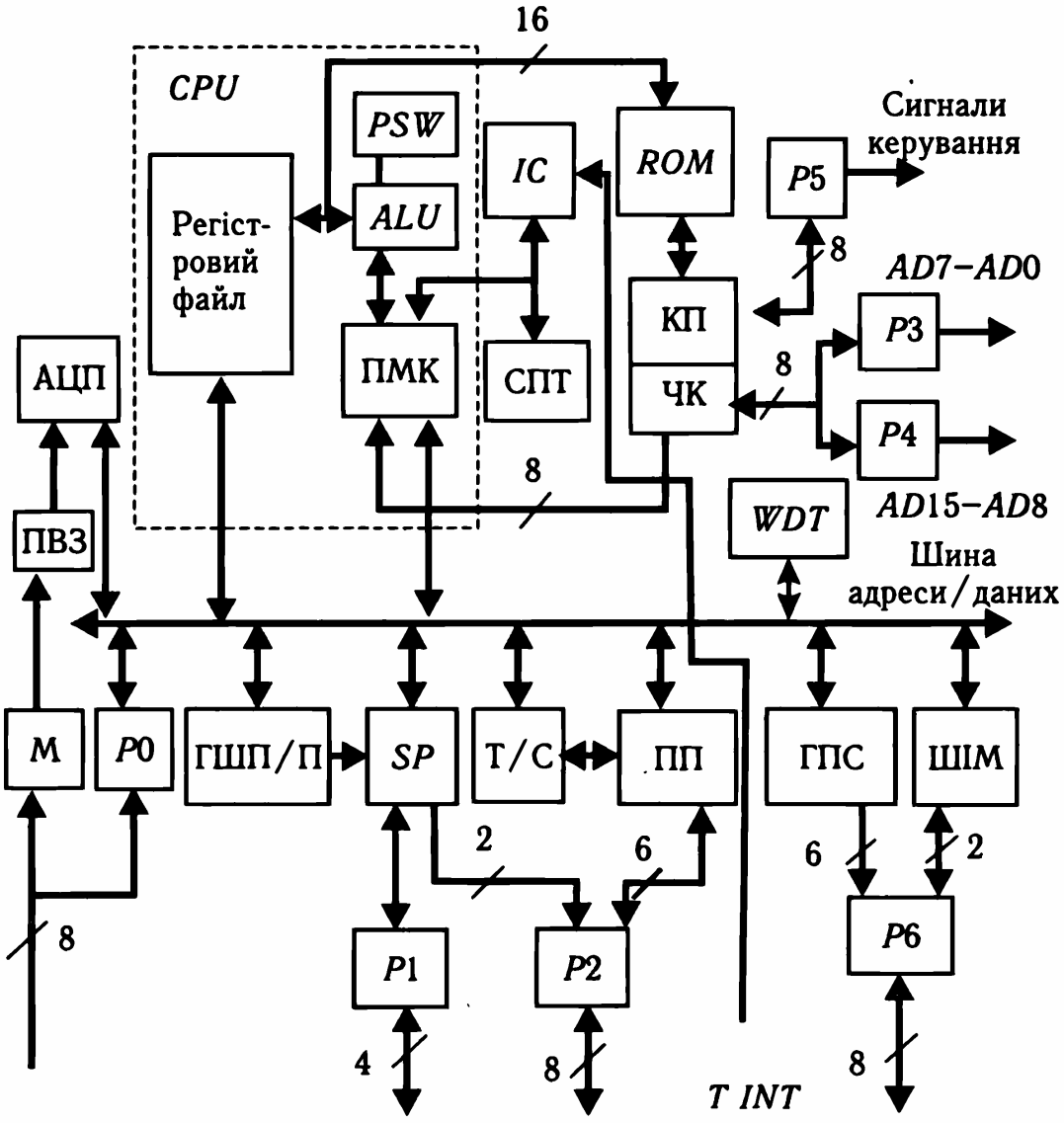


Рис. 6.30. Структурна схема ОМК з інтегрованими засобами керування двигунами

Генератор періодичних сигналів містить:

• блок цифрового опорного генератора (*Timebase Generator*), що задає пилкоподібну або трикутну форму опорного цифрового сигналу модулятора;

• блок триканального цифрового компаратора — драйвера формування сигналів фаз (*Phase Driver*), в якому на базі заданого значення шпаруватості формується три незалежних ШІМ-сигнали — *WG*1, *WG*2, *WG*3. Цей блок формує також три інверсних сигнали — *WG*1#, *WG*2#, *WG*3# з урахуванням затримки ввімкнення для уникнення режиму наскрізного струму;

• блок керування виходами генератора періодичних сигналів (*Output Control*), який дозволяє програмно ввімкнути чи вимкнути будь-який з сигналів *WGn* або *WG#n*, призначити для виходів генератора активний рівень (*L* або *H*), перевести за інтервал часу менше 2 мкс виходи генератора у пасивний стан після надходження сигналу аварії, вимкнути всі ШІМ-виходи програмно.

Генератор періодичних сигналів дає змогу у широких межах регулювати частоту і період ШІМ (від 0,25 мкс до 16 мс для односторонньої ШІМ і від 0,125 мкс до 8 мс — для двосторонньої). Тривалість затримки для усунення наскрізного струму становить 0,125—125 мкс

Однокристальний мікроконтролер з інтегрованими засобами цифрового оброблення. Однокристальний мікроконтролер MCS296 має традиційну архітектуру Intel процесорів оброблення подій у реальному масштабі часу, але додатково містить цифровий сигнальний процесор *Digital Signal Processor* (*DSP*) для реалізації функції цифрового регулятора. Таке поєднання різних типів процесорів дало змогу отримати високі технічні характеристики конторолерів:

• підвищення продуктивності ЦП за рахунок зменшення тривалості вибирання з пам'яті програм і даних;

• використання конвеєрного режиму з одночасним виконанням чотирьох команд для зменшення тривалості машинного циклу;

• застосування апаратних засобів множення і ділення;

• доповнення системи команд сімнадцятьма спеціальними командами (наприклад, множення з накопиченням, повторення, автоматизація оновлення даних у таблицях послідовних вибірок вхідних сигналів у процесі їх оброблення), оптимізованими для розв'язання задач побудови цифрових фільтрів і регуляторів.

Контрольні запитання

1. Назвіть основні складові блока ОМК.
2. Поясніть структуру та принцип функціонування блока керування ОМК.
3. Як здійснюється синхронізація роботи основних блоків ОМК?
4. Яке призначення та характеристики РПП?
5. Яке призначення та характеристики РПД?
6. Яке призначення та режими роботи таймерів?
7. Яке призначення та режими роботи портів ОМК?
8. Яке призначення та режими роботи послідовного порту ОМК?
9. Яке призначення системи переривань і джерела переривань в ОМК?
10. Поясніть принцип пріоритетного оброблення запитів переривань.
11. Які є режими роботи послідовного порту?
12. Чим відрізняється прапорець нуля в ОМК К1816ВЕ51 та МП *i*8086?
13. Які є групи команд?
14. Які можливі операції над бітами?
15. Які арифметичні операції можна здійснювати в ОМК К1816ВЕ51?
16. Які логічні операції можна здійснювати в ОМК К1816ВЕ51?
17. Скільки є байт з прямою адресацією?
18. З якими форматами даних оперує система команд ОМК К1816ВЕ51?
19. Як здійснюється розширення пам'яті програм в ОМК?
20. Як здійснюються звернення до внутрішньої і зовнішньої пам'яті програм?
21. За допомогою яких портів здійснюється передавання адреси комірки зовнішньої пам'яті програм?
22. Як здійснюється розширення пам'яті даних в ОМК?
23. За допомогою яких портів здійснюється передавання адреси комірки зовнішньої пам'яті даних?
24. Скільки додаткових портів введення-виведення можна з'єднати з ОМК?
25. Наведіть основні характеристики ОМК 83С51FA.
26. Які особливості режиму захоплення події?
27. Назвіть особливості режиму широтно-імпульсного модулятора.
28. Назвіть особливості режиму watchdog-таймера.
29. Які події можуть бути зафіксовані в режимі захоплення події?
30. Поясніть принцип формування сигналу широтно-імпульсного модулятора на виводі ОМК.
31. Дайте визначення події, захоплення події та трансакції.
32. Яку інформацію містить регістровий файл в ОМК з модулем *HSIO*?
33. Назвіть призначення та поясніть принцип дії сервера периферійних трансакцій.
34. Назвіть призначення та поясніть принцип дії модуля високошвидкісного введення-виведення.
35. Який принцип роботи вбудованого АЦП в ОМК з модулем *HSIO*?
36. Порівняйте характеристики послідовного порту ОМК з ВІС КР580ВВ51.
37. Порівняйте характеристики блока таймерів *EPA*-мікроконтролера і *HSIO*-мікроконтролера.
38. Які особливості квадратурного режиму роботи таймера?
39. Порівняйте характеристики генератора періодичних сигналів мікроконтролера з модулем *HSIO*- і *MCF*-мікроконтролерів.

Розділ 7 ОДНОКРИСТАЛЬНІ МІКРОКОНТРОЛЕРИ З RISC-АРХІТЕКТУРОЮ

CISC мікроконтролери характеризуються досить розвиненою системою, наприклад, мікроконтролери серії *i*80 × 51 мають 111 команд. Однак аналіз програм показав, що 20 % команд використовуються у 80 % випадків, а дешифратор команд займає понад 70 % усієї площі кристала. Тому у розробників МП виникла ідея скоротити кількість команд, придати їм єдиний формат і зменшити площу кристала, тобто використати RISC *(Reduced Instruction Set Computer)* архітектуру.

Особливістю контролерів, виконаних за RISC архітектурою, є те, що всі команди виконуються за один-три такти, тоді як у CISC контролерах — за один-три машинних цикли, кожний з яких складається з кількох тактів, наприклад для *i*80 × 51 — з 12 тактів. Тому RISC-контролери мають значно більшу швидкодію. Однак повніша система команд CISC контролерів у деяких випадках сприяє економії часу виконання певних фрагментів програми та економії пам'яті програм.

7.1. PIC-Контролери

Типовими представниками RISC-процесорів є PIC-контролери (*Periferial Interface Controller* — контролери периферійних інтерфейсів) виробництва фірми Microchip. PIC-Контролери застосовують у системах високошвидкісного керування автомобільними й електричними двигунами, приладах побутової електроніки, телефонних приставках з АВН, системах охорони із сповіщанням по телефонній лінії, міні-АТС. Окремі ВІС відрізняються розрядністю ПЗП: від 12 до 14 біт для серії PIC16Cxx та 16 біт для серії PIC17Cxx. Завдяки скороченій кількості команд (від 33 до 35) усі команди займають у пам'яті одне слово. Тривалість виконання кожної

команди, крім команд розгалуження, становить чотири такти — один цикл (200 нс за частоти 20 МГц). Оперативний запам'ятовувальний пристрій виконано за схемою з довільною вибіркою та можливістю безпосередньої адресації у коді команди для будь-якої комірки. Стек реалізовано апаратно з глибиною 2, 8 або 16 комірок. Майже в усіх PIC-контролерах є система переривань, джерелом яких може бути таймер, а також зміна станів сигналів на деяких входах. У PIC-контролерах передбачено біт захисту ПЗП, що запобігає нелегальному копіюванню. Основні характеристики різних типів PIC-контролерів наведено у табл. 7.1.

Великі інтегральні схеми PIC16Cxx мають вбудовані ПЗП ємністю від 0,5 до 4 кілослів і ОЗП ємністю 32—256 байт. Основна частина контролерів має однократно програмований ПЗП, однак деякі контролери містять ПЗП з ультрафіолетовим стиранням, а PIC 16C84 містить пам'ять програм та пам'ять даних на базі ПЗП з електричним стиранням. Крім того, контролери мають від одного до трьох таймерів, вбудовану систему скидання, watchdog-таймер, внутрішній тактовий генератор, який запускається як від кварцового резонатора, так і від RC-ланцюга у широкому діапазоні частот — від 0 до 25 МГц. Кількість розрядів портів становить 12—33. Кожний розряд порту можна запрограмувати на введення або на виведення. Контролер PIC 16C64 додатково має вихід із ШІМ, за допомогою якого можна реалізувати ЦАП з розрядністю до 16 розрядів, та послідовний двонапрямлений синхронний порт з інтерфейсами SPI, I2C, SCI/UART. Контролери PIC 16C71 і PIC 16C74 мають внутрішній 8-розрядний АЦП з пристроєм вибирання-зберігання і вхідним аналоговим мультиплексором.

Контролери PIC17Cxx мають вбудований апаратний 8-розрядний помножувач, два виходи ШІМ з роздільною здатністю 1...10 біт, два виводи з відкритим колектором, чотири таймери-лічильники, одинадцять джерел переривань, у них передбачено можливість виконання програми із зовнішнього ПЗП.

Контролер PIC 1400 має програмований вибір генератора — вбудованого резонатора з частотою 4 МГц або зовнішнього кварцового чи керамічного резонатора, сторожовий таймер з окремо вбудованим RC-генератором, внутрішньосхемним програмуванням через два виводи, два режими зниженого енергоспоживання (струм 200 мкА, напруга живлення 3В за вимкненого генератора й активних аналогових схемах; струм 5 мкА, напруга живлення 3 В за вимкненого генератора й вимкнених аналогових схемах).

Таблиця 7.1. **Основні технічні характеристики PIC-процесорів**

Тип PIC	Технічні характеристики														
	Тактова частота, МГц	Пам'ять програм, біт		Пам'ять даних, байт	EEPROM пам'ять даних, байт	Глибина стеку	Таймер 0 (8 + 8 біт)	Таймер 1 (16 біт)	Таймер 2 (8 біт)	Кількість ШІМ виходів	Послідовний порт-інтерфейс	Кількість каналів АЦП	Кількість запитів переривань	Кількість розрядів введення-виведення	Кількість контактів ВІС
		EPROM	EEPROM												
16C54	16	512 × 12	–	36	–	2	+	–	–	–	–	–	–	13	18
16C55	16	512 × 12	–	36	–	2	+	–	–	–	–	–	–	21	28
16C56	16	1024 × 12	–	72	–	2	+	–	–	–	–	–	–	13	18
16C57	16	2048 × 12	–	72	–	2	+	–	–	–	–	–	–	21	28
16C58	16	2048 × 12	–	80	–	8	+	–	–	–	–	–	–	13	18
16C61	20	–	1024 × 14	36	–	8	+	–	–	–	–	–	3	13	18
16C62	20	2048 × 14	–	128	–	8	+	+	+	1	SPI/I2C, SCI/UART	–	10	21	28
16C620	20	–	512 × 14	80	–	8	+	–	–	–	–	–	4	13	18
16C621	20	–	1024 × 14	80	–	8	+	–	–	–	–	–	4	13	18
16C622	20	–	2048 × 14	128	–	8	+	–	–	–	–	–	4	13	18

16C63	20	4096 × 14	–	192	–	8	+	+	+	1	SPI/I2C, SCI/UART	–	+	21	28
16C64	25	2048 × 14	–	128	–	8	+	+	+	1	SPI/I2C, SCI/UART	–	8	21	40
16C65	20	4096 × 14	–	192	–	8	+	+	+	2	SPI/I2C, SCI/UART	–	8	21	40
16C710	20	–	1512 × 14	36	–	8	+	–	–	–	–	4	3	13	18
16C71	16	–	1024 × 14	36	–	8	+	–	–	–	–	4	3	13	18
16C711	20	–	1024 × 14	68	–	8	+	–	–	–	–	4	3	13	18
16C72	20	2048 × 14	–	128	–	8	+	+	+	1	SPI/I2C, SCI/UART	5	10	21	28
16C73	20	4096 × 14	–	192	–	8	+	+	+	2	SPI/I2C, SCI/UART	5	10	21	28
16C74	20	4096 × 14	–	192	–	8	+	+	+	2	SPI/I2C, SCI/UART	8	8	21	40
16C83	10	512 × 14	64 × 8	64	36 × 8	8	+	–	–	–	–	–	–	13	18
16C84	10	1024 × 14	64 × 8	64	36 × 8	8	+	–	–	–	–	–	–	13	18
16C84A	10	1024 × 14	64 × 8	64	68 × 8	8	+	–	–	–	–	–	–	13	18
1400	20	4096 × 14	–	192	–	8	+	+	–	–	SPI/I2C,	8	–	21	28
17C42	25	–	2048 × 16	232	–	16	+	+	+	2	*SCI/UART*	–	+	32	40
17C43	25	–	4096 × 16	454	–	16	+	+	+	2	SCI/UART	–	+	32	40
17C44	25	–	8192 × 16	454	–	16	+	+	+	2	SCI/UART	–	+	32	40

Процесор PIC 1400 містить інтегральний АЦП на вісім каналів, діапазон напруги двох з яких може задаватися програмно. Тривалість перетворення АЦП становить 16 мс за тактової частоти 4 МГц і роздільної здатності 16 біт. Цей контролер містить також чотирирозрядний ЦАП струму, вбудований датчик температури з роздільною здатністю 0,10 В та вбудований детектор зниженої напруги живлення.

Архітектура PIC-контролерів. Архітектуру PIC-контролерів розглянемо на прикладі BIC PIC16C71 (рис. 7.1). Основою архітектури є роздільні шини та області пам'яті для даних і команд. Шина даних та комірка ОЗП є 8-розрядними, а шина команд і програмна пам'ять (ПЗП) — 14-розрядними; 14-розрядна команда вибирається за один цикл. Двосхідчастий конвеєр забезпечує одночасне вибирання і виконання команди. Система команд містить 35 команд. Усі команди виконуються за один цикл, за винятком команд переходів, що виконуються за два цикли.

Структурна схема контролера містить:

- восьмирівневий апаратний стек;
- 13-розрядний програмний лічильник *PC*;
- 8-розрядний АЛП;
- ОЗП, який складається з 36 8-розрядних РЗП;
- 15 регістрів спеціальних функцій *SFR* (на рис. 7.1 показано регістр непрямої адресації *W* та регістри статусу, опцій і конфігурації ПЗП);

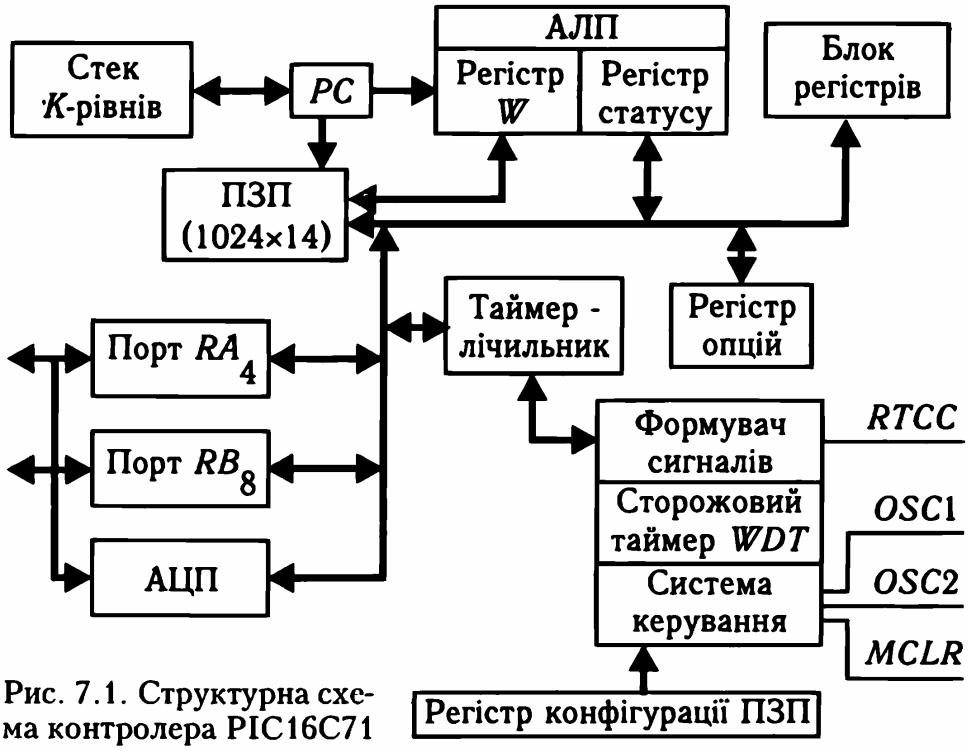


Рис. 7.1. Структурна схема контролера PIC16C71

• 8-розрядний таймер-лічильник з 8-розрядним програмним передподільником;
• модуль АЦП з чотирма входами;
• 13 ліній введення-виведення (чотирирозрядний порт *RA*, 8-розрядний порт *RB*, лінія *RTCC*);
• сторожовий таймер;
• формувач зовнішнього сигналу *RTCC* або сигналу сторожового таймера;
• система керування і синхронізації з внутрішнім генератором.

Призначення виводів ВІС наведено у табл. 7.2, а граничні електричні параметри — у табл. 7.3.

Таблиця 7.2. **Призначення виводів PIC16C71**

Номер виводу	Позначення	Функціональне призначення
1	*RA2/AIN2*	Двонапрямлена лінія введення-виведення. Аналоговий вхід каналу 2. Як цифровий вхід має рівні ТТЛ
2	*RA3/AIN3/Vref*	Двонапрямлена лінія введення-виведення. Аналоговий вхід каналу 3
3	*RA4/RTCC*	Вхід через тригер Шмітта. Лінія порту введення-виведення з відкритим стоком або вхід частоти для таймера лічильника *RTCC*
4	$\overline{MCLR}/U_{pp}$	Сигнал скидання для контролера. Активний низький рівень. Вхід через тригер Шмітта
5	*Uss*	Загальний вивід (0 В)
6	*RB0/INT*	Двонапрямлена лінія введення-виведення або зовнішній вхід переривання
7	*RB1*	Двонапрямлена лінія введення-виведення
8	*RB2*	Те саме
9	*RB3*	– « –
10	*RB4*	– « –
11	*RB5*	– « –
12	*RB6*	– « –
13	*RB7*	– « –
14	*Udd*	Напруга живлення
15	*OSC2/CLKOUT*	Вхід увімкнення кварцового резонатора в усіх режимах, крім RC-генератора (вихід тактової частоти в режимі RC-генератора)

Продовж. табл. 7.2

Номер виводу	Позначення	Функціональне призначення
16	*OSC1/CLKIN*	Вхід ввімкнення кварцового резонатора, RC-ланцюга (вхід зовнішньої тактової частоти)
17	*RA0/AIN0*	Двонапрямлена лінія введення-виведення. Аналоговий вхід каналу 0
18	*RA1/AIN1*	Двонапрямлена лінія введення-виведення. Рівні ТТЛ. Аналоговий вхід каналу 1

Таблиця 7.3. **Граничні значення електричних параметрів**

Параметр	Граничне значення
Інтервал робочих температур	−55...+125 °С
Температура зберігання	−65...+150 °С
Напруга на будь-якому виводі щодо *Uss* (землі),крім *Udd* і */MCLR*	−0,6... *Udd* + 0,6 В
Напруга *Udd* щодо *Uss*	0...7,5 В
Напруга */MCLR* щодо *Uss*	0... 14 В*
Загальна потужність розсіювання	800 мВт
Максимальний струм на виводі *Uss*	150 мА
Максимальний струм на виводі *Udd*	100 мА
Максимальний струм на будь-якому виводі	± 500 мкА
Максимальний вхідний струм будь-якої лінії порту *RA* або *RB*	25 мА
Максимальний вихідний струм будь-якої лінії порту *RA* або *RB*	20 мА
Максимальний сумарний вхідний струм для всіх ліній порту *RA*	50 мА
Максимальний сумарний вхідний струм для всіх ліній порту *RB*	100 мА
Максимальний сумарний вихідний струм для всіх ліній порту *RA*	80 мА
Максимальний сумарний вихідний струм для всіх ліній порту *RB*	150 мА

Примітка.Сигнал на вивід */MCLR* рекомендується подавати через обмежувальний резистор 50...100 Ом.

Як видно з наведених характеристик, PIC-контролери за своїми параметрами конкурують з однокристальними мікроЕОМ та ОМК. Деякі модифікації PIC-контролерів мають більшу швидкодію, ніж ОМК. PIC-контролери та ОМК застосовують у вбудованих системах керування різного призначення.

7.2. Однокристальні AVR-мікроконтролери

Однокристальні AVR-мікроконтролери — це 8-розрядні високопродуктивні RISC-контролери загального призначення. Вони були створені групою розроблювачів дослідницького центру фірми Atmel Corp. (Норвегія), ініціали яких сформували марку AVR. Особливістю AVR-мікроконтролерів є їх широка номенклатура, що дає змогу користувачу вибрати мікроконтролер з мінімальною апаратною надлишковістю і, отже, найменшої вартості. Так, у номенклатуру групи AT90S входять прилади з ПЗП ємністю від 1 до 8 Кбайт із різними наборами периферії у корпусах із кількістю виводів від 8 до 48. Нині у серійному виробництві знаходяться три сімейства AVR — Tiny, Classic і Mega. Мікроконтролери Tiny — найдешевші і мають найпростішу структуру, Mega — найпотужніші, а Classic — займають проміжне положення між ними.

Розглянемо архітектуру AVR-мікроконтролерів на прикладі МК сімейства Classic — AT90S8535 (рис. 7.2)

Мікроконтролер містить гарвардський процесор, регістровий файл, пам'ять програм, пам'ять даних та різні інтерфейсні схеми (периферію).

Гарвардський процесор реалізує повний логічний і фізичний розподіл не лише адресних просторів, а й інформаційних шин для звертання до пам'яті програм і до пам'яті даних, причому способи адресування й доступу до цих масивів пам'яті також різні.

Подібна побудова ближча до структури цифрових сигнальних процесорів і забезпечує істотне підвищення продуктивності. Процесор працює одночасно як із пам'яттю програм, так і з пам'яттю даних; розрядність шини пам'яті програм розширена до 16 біт. У МК AVR використовується технології конвеєризації, внаслідок чого цикл вибірка-виконання команди значно скорочений. Для порівняння, у мікроконтролерів сімейства MCS51 коротка команда виконується за 12 тактів генератора (один машинний цикл), упродовж якого процесор послідовно зчитує код операції і виконує її. У PIC-кон-

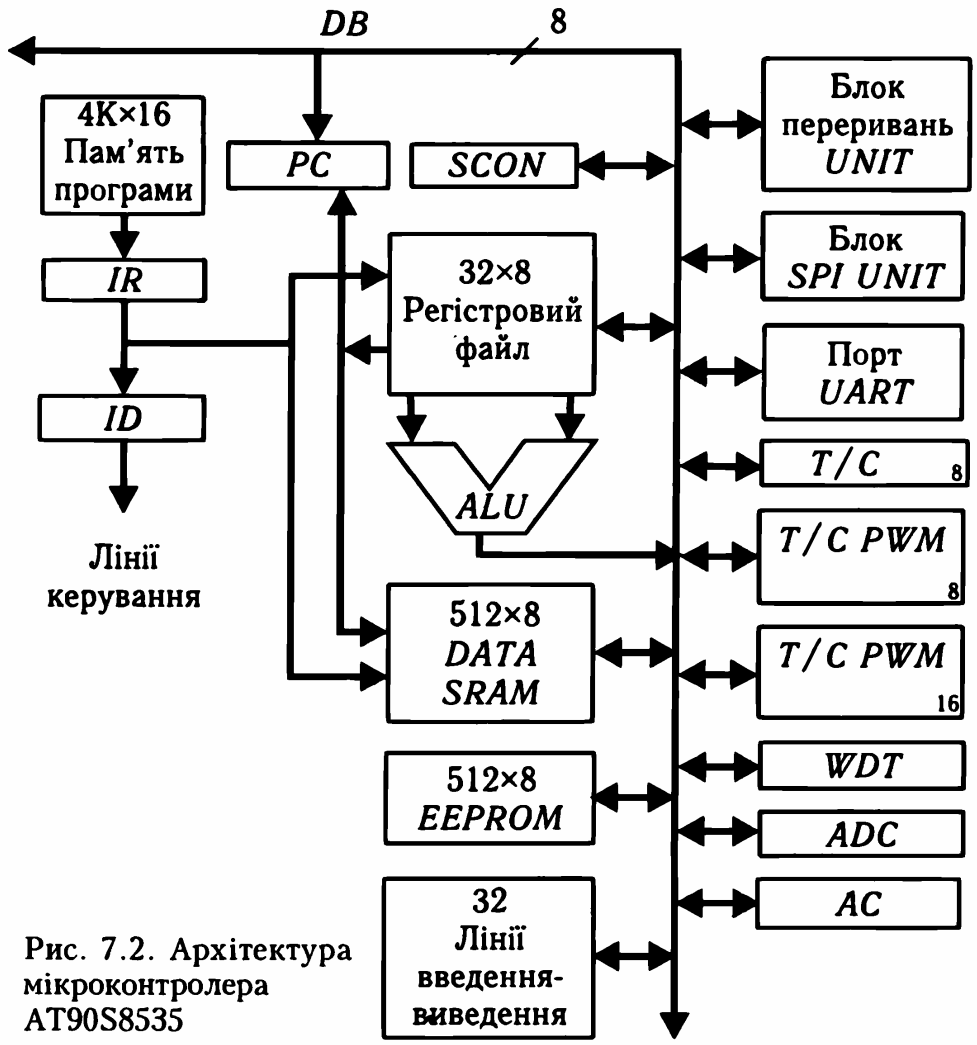
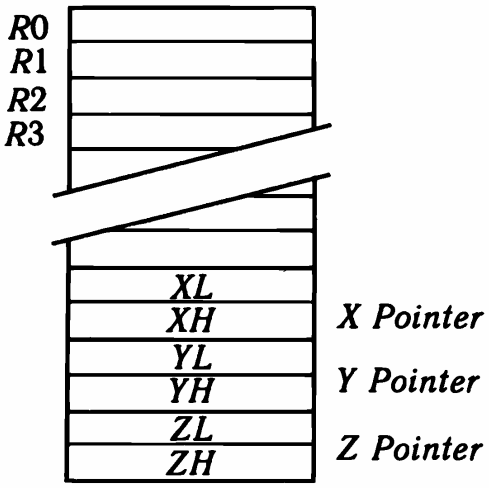


Рис. 7.2. Архітектура мікроконтролера AT90S8535

тролерах фірми Microchip, де вже реалізований конвеєр, коротка команда виконується впродовж 8 періодів тактової частоти (2 машинних цикли). За цей час послідовно дешифрується і зчитується код операції, здійснюється команда, фіксується результат й одночасно зчитується код наступної операції (однорівневий конвеєр). Тому в загальному потоці команд одна коротка команда реалізується за чотири періоди тактової частоти або за один машинний цикл. У МК AVR також використовується однорівневий конвеєр під час звертання до пам'яті програм і короткі команди в загальному потоці виконуються, як і в PIC-контролерах, за один машинний цикл. Головна відмінність полягає в тому, що цей цикл у МК AVR становить усього один період тактової частоти.

Регістровий файл займає молодші 32 байти в загальному адресному просторі SRAM AVR (рис. 7.3). Шість із 32-х регістрів файла можуть використовуватися як три 16-розрядних покажчики адреси за непрямого адресування даних. Один з цих покажчиків (*Z Pointer*) застосовується також

Рис. 7.3. Регістровий файл



для доступу до даних, записаних у пам'яті програм мікроконтролера. Використання трьох 16-бітних покажчиків (*X, Y і Z Pointers*) значно підвищує швидкість пересилання даних під час роботи прикладної програми.

Пам'ять програм. Усі AVR- мікроконтролери мають Flash-пам'ять програм, що може бути завантажена як за допомогою звичайного програматора, так і за допомоою SPI-інтерфейсу, у тому числі безпосередньо на цільовій платі. Число циклів перезапису — не менше 1000. Деякі версії кристалів сімейства Mega мають можливість самопрограмування, тобто мікроконтролер здатний самостійно (без зовнішнього програматора) змінювати вміст комірок пам'яті програм. Це дає змогу записати у зовнішню енергонезалежну пам'ять кілька робочих версій програми, а потім у міру потреби або за реакцією на зовнішні або внутрішні логічні умови перевантажувати робочі програми у той самий мікроконтролер AVR без витягування його з друкарської плати. Для цього весь масив пам'яті програм поділяється на дві нерівні по ємності області: блок завантажувача (програма, що керує перезаписом Flash-пам'яті програм) і блок для розміщення робочих програм. Програма-завантажувач створюється самим розроблювачем і має бути запрограмована зовнішнім програматором.

Пам'ять даних. Усі AVR-мікроконтролери мають також блок енергонезалежної пам'яті даних з електричним стиранням EEPROM. Цей тип пам'яті використовується для збереження проміжних даних, різних констант, таблиць перекодувань, каліброваних коефіцієнтів тощо. Дані в EEPROM можуть бути завантажені як через SPI інтерфейс, так і за допомогою звичайного програматора. Число циклів перезапису — не менш як 100 000. Два програмні біти захисту інформації дають змогу захистити пам'ять програм та енергонезалежної пам'яті даних EEPROM від несанкціонованого зчитування.

Внутрішня оперативна пам'ять SRAM є в усіх AVR сімейств Classic та Mega і в одного нового кристала сімейства Ttiny — ATtiny26/L. Для деяких мікроконтролерів можлива організація підключення зовнішньої пам'яті даних ємністю до 64 Кбайт.

Периферія МК AVR. До числа старих периферійних пристроїв належать 8-розрядні порти введення-виведення, послідовний порт, таймери-лічильники, контролер переривань.

Число незалежних ліній портів введення-виведення становить від 3 до 53. Кожний розряд порту може бути запрограмований на введення або виведення інформації. Потужні вихідні драйвери забезпечують навантажувальну спроможність струму 20 мА на лінію порту за максимального значення 40 мА, що дає змогу, наприклад, безпосередньо підключати до мікроконтролера світлодіоди і біполярні транзистори. Загальне навантаження струму на всі лінії одного порту має не перевищувати 80 мА (усі значення наведені для напруги живлення 5 В).

Мікроконтролери AVR мають у своєму складі від 1 до 4 таймерів-лічильників загального призначення з розрядністю 8 або 16 біт, що можуть працювати і як таймери від внутрішнього джерела опорної частоти, і як лічильники зовнішніх подій із зовнішнім тактуванням.

Таймер-лічильник *RTC* (є в усіх МК сімейства Mega та в деяких МК Classic) реалізує систему реального часу. Таймер має свій власний передподільник, що може бути програмно підключений або до основного внутрішнього джерела тактової частоти мікроконтролера, або до додаткового асинхронного джерела опорної частоти (кварцовий резонатор або зовнішній синхросигнал). Тому МК має два зовнішніх виводи. Внутрішній осцилятор навантажений на лічильний вхід таймера-лічильника *RTC* та оптимізований для роботи із зовнішнім «годинним» кварцовим резонатором 32,768 кГц.

Новими периферійними пристроями є блок послідовного периферійного інтерфейсу (*SPI*), сторожовий таймер (*WDT*) та аналоговий компаратор (*AC*).

Блок *SPI* призначений для послідовного введення-виведення даних і використовується для програмування мікроконтролера після встановлення його на друкарську плату.

Сторожовий таймер *WDT* (*WATCHDOG*) призначений для перезапуску програми у разі збою у ході її виконання. Програма, що працює без збоїв, періодично скидає сторожовий таймер, не допускаючи його переповнювання. Сторожовий таймер має свій власний RC-генератор, що працює на частоті 1 МГц. На вході *WDT* підключений передподільник вхідної частоти з програмним коефіцієнтом ділення, що дає змогу регулювати часовий інтервал переповнення таймера і скидання мікроконтролера. Таймер *WDT* може бути відключений програмним способом під час роботи мікроконтролера як в активному режимі, так і в будь-якому з режимів знише-

ного енергоспоживання. В останньому випадку це сприяє значному зниженню споживаного струму.

Аналоговий компаратор *AC* порівнює за величиною напруги сигнали, що надходять на входи *P*1.0 і *P*1.1. Результат порівняння подається на вхід *P*3.6, що не має зовнішнього виводу.

Аналого-цифровий перетворювач *ADC* побудований за класичною схемою послідовних наближень з пристроєм вибірки-збереження (ПВЗ). Кожний з аналогових входів може бути з'єднаний з входом ПВЗ через аналоговий мультиплексор. Пристрій вибірки-збереження має свій власний підсилювач, який гарантує, що вимірюваний аналоговий сигнал буде сталим упродовж усього часу перетворення. Розрядність АЦП становить 10 біт за нормованої похибки ±2 розряди. АЦП може працювати у двох режимах — однократне перетворення на будь-якому обраному каналі та послідовне циклічне опитування усіх каналів. Тривалість перетворення вибирається програмно за допомогою встановлення коефіцієнта розподілу частоти спеціального передподільника, що входить до складу блока АЦП. Воно становить 70...280 мкс для мікроконтролерів ATmega103 і 65...260 мкс для всіх інших мікроконтролерів, що мають у своєму складі АЦП.

Внутрішній тактовий генератор МК AVR може запускатися від кількох джерел опорної частоти (зовнішній генератор, зовнішній кварцовий резонатор, внутрішній або зовнішній RC-ланцюг). Оскільки AVR-мікроконтролери цілком статичні, мінімальна допустима частота нічим не обмежена, тобто можливо легко забезпечити навіть покроковий режим виконання програми. Максимальна робоча частота визначається конкретним типом мікроконтролера.

Мікроконтролери AVR можна перевести програмно в один із шести режимів зниженого енергоспоживання:

- режим холостого ходу (*IDLE*), в якому припиняє роботу лише процесор і фіксується вміст пам'яті даних, а внутрішній генератор синхросигналів, таймери, система переривань і *WDG*-таймер продовжують функціонувати.
- режим мікроспоживання (*Power Down*), в якому зберігається вміст регістрового файла, але зупиняється внутрішній генератор синхросигналів. Вихід із *Power Down* можливий або після загального скидання мікроконтролера, або після сигналу від зовнішнього джерела переривання. Для ввімкнення *WDG*-таймера струм споживання в цьому режимі становить 60...80 мкА, а для вимкненого — менше ніж 1 мкА для всіх типів AVR. Наведені вище значення справедливі для напруги живлення 5 В;
- режим зберігання енергії (*Power Save*), що реалізований лише у тих AVR, що мають у своєму складі систему реально-

го часу. Режим *Power Save* ідентичний режиму *Power Down*, але він допускає незалежну роботу таймера/лічильника *RTC*. Вихід із режиму *Power Save* можливий після переривання, зумовленому або переповненням таймера-лічильника *RTC*, або спрацьовування блока порівняння цього лічильника. Струм споживання у цьому режимі становить 6...10 мкА за напруги живлення 5 В на частоті 32,768 кГц;

• режим заглушення шуму в процесі роботи аналого-цифрового перетворювача (*ADC Noise Reduction*). У цьому режимі припиняється робота процесора, але дозволяється робота АЦП, двопровідного інтерфейсу I^2C і сторожового таймера.

• основний режим очікування (*Standby*). Відрізняється від режиму *Power Down* тим, що робота тактового генератора не припиняється. Це гарантує швидкий вихід мікроконтролера з режиму очікування всього за 6 тактів генератора;

• додатковий режим очікування (*Extended Standby*) ідентичний режиму *Power Save*, але робота тактового генератора також не припиняється і гарантує швидкий вихід з режиму за 6 тактів генератора.

Система команд AVR налічує до 133 різних інструкцій. Розрізняють п'ять груп команд AVR: умовного розгалуження, безумовного розгалуження, арифметичні та логічні операції, команди пересилання даних, команди роботи з бітами. В останніх розробках AVR сімейства Mega реалізована функція апаратного множення. За кількістю реалізованих конструкцій AVR-мікроконтролери більше на CISC, ніж на RISC-процесори. У PIC-контролерів система команд налічує подібних до 75 різних інструкцій, а в MCS51 вона становить 111.

У цілому прогресивна RISC архітектура AVR у поєднанні з наявністю регістрового файла і розширеної системи команд дає змогу створювати компактні програми з високою швидкістю виконання.

7.3. Характеристики AVR-мікроконтролерів

Докладніше розглянемо характеристики мікроконтролерів сімейств AVR:

• Classic AVR — основне сімейство мікроконтролерів продуктивністю до 16 MIPS*, пам'ять програм FLASH ROM 2—8 Кбайт, пам'ять даних EEPROM 64—512 байт, пам'ять даних SRAM 128—512 байт;

*MIPS — мільйон операцій за секунду.

• Mega AVR продуктивністю 4—6 MIPS, пам'ять програм FLASH ROM 64—128 Кбайт, пам'ять даних EEPROM 64—4096 байт, пам'ять даних SRAM 1—4 Кбайт, вбудований 10-розрядний 8-канальний АЦП, апаратний помножувач 8×8;

• Tiny AVR — низьковартісні мікроконтролери у 8-вивідному виконанні, мають вбудовану схему контролю напруги живлення.

Мікроконтролери Tiny характеризується найменшою серед AVR МК ємністю пам'яті програм та обмеженим набором функцій. Однак малогабаритні корпуси, можливість роботи за напруги живлення 1,8 В (МК з індексом *U*) дають змогу використовувати ці мікроконтролери у портативній апаратурі, зокрема з батарейним живленням.

За рівнем інтеграції і можливостей найпотужнішими є МП групи Mega. Для мікроконтролерів цієї групи, особливо для недавно розроблених, характерні:

• велика ємність Flash-пам'яті програм (від 8 до 128 Кбайт);

• режим самопрограмування, забезпечений вбудованою програмою-завантажником;

• вбудований помножувач, що підтримує множення дробових чисел зі знаком та без знака;

• розширені набори вбудованої периферії;

• широкий набір спеціальних функцій мікроконтролера, у тому числі до шести режимів енергозбереження і можливість програмного встановлення тактової частоти;

• розширення системи команд до 130—133, у тому числі кількома командами 32-розрядного формату;

• організація в нових моделях інтерфейсу граничного сканування (*IEEE* 1149.1/*JTAG*), що підтримує вбудоване налагодження і забезпечує ще один спосіб програмування Flash- і EEPROM-пам'яті, перемичок і бітів блокування;

• спеціальні функції мікроконтролера, що забезпечують високу усталеність роботи апаратних і програмних засобів у разі випадкових змін напруги живлення.

Галузі застосування мікроконтролерів цієї групи — від високонадійних функцій традиційного обчислення і керування до оброблення сигналів і керування двигунами.

Сімейство мікроконтролерів Classic нині найпоширеніше. Мікроконтролери цього класу мають менші периферію та обчислювальні можливості, ніж контролери сімейства Mega, але більші, ніж сімейства Tiny. Корпорація Atmel не планує подальший розвиток МК сімейства Classic, оскільки вважається, що це сімейство функціонально збалансоване і досить різноманітне за функціональними можливостями.

У табл. 7.4.—7.6. наведено характеристики мікроконтролерів відповідно сімейств Tiny, Classic і Mega. В цих таблицях такі позначення:

- *Flash ROM* — ємність енергонезалежної пам'яті програм (у кілобайтах);
- *EEPROM* — ємність енергонезалежної пам'яті даних (у байтах);
- *SRAM* — ємність статичної пам'яті даних (у байтах);
- *ISP* — можливість програмування мікроконтролера на цільовій платі (*I*) за основної напруги живлення або можливість самопрограмування без участі зовнішнього програматора (*S*);
- *I/O* — кількість ліній введення-виведення;

Таблиця 7.4. **Мікроконтролери AVR сімейства Tiny**

Тип	Технічні характеристики									
	Напруга живлення, В	Тактова частота, МГц	I/O	Flash	EEPROM	SRAM	Інтерфейси	Аналогові входи	Таймери	ISP
AT-tiny 11L	2,7...5,5	2	6	1K	—	—	—	—	1×8	—
AT-tiny 11	4,0...5,5	6	6	1K	—	—	—	—	1×8	—
AT-tiny 12V	1,8...5,5	1	6	1K	64	—	—	—	1×8	I
AT-tiny 12L	2,7...5,5	4	6	1K	64	—	—	—	1×8	I
AT-tiny 15L	2,7...5,5	1	6	1K	64	—	—	ADC 4×10	2×8	I
AT-tiny 26L	2,7...5,5	8	16	1K	128	128	SPIU ART	ADC 11×10	2×8	I
AT-tiny 26	4,0...5,5	16	16	1K	128	128	SPIU ART	ADC 11×10	2×8	I
AT-tiny 28V	1,8...5,5	1	20	2K	—	—	—	—	1×8	—
AT-tiny 28L	2,7...5,5	4	20	2K	—	—	—	—	1×8	—

Таблиця 7.5. **Мікроконтролери AVR сімейства Classic**

Тип	Технічні характеристики									
	Напруга живлення, В	Тактова частота, МГц	I/O	Flash	EEPROM	SRAM	Інтерфейси	Аналогові входи	Таймери	ISP
AT90S12	2,7...6,0 4,0...6,0	4 12	15	1K	64	–	–	–	1×8 1×16	I
AT90S23	2,7...6,0 4,0...6,0	4 10	15	2K	128	128	UART	–	1×8	I
AT90L2323	2,7...6,0	4	3	2K	128	128	–	–	1×8	I
AT90S2323	4,0...6,0	10	3	2K	128	128	–	–	1×8	I
AT90LS2343	2,7...6,0	4	5	2K	128	128	–	–	1×8	I
AT90S2343	4,0...6,0	10	5	2K	128	128	–	–	1×8	I
AT90LS4433	2,7...6,0	4	20	4K	256	128	UART SPI	ADC 6×10	1×8 1×16	I
AT90S4433	4,0...6,0	8	20	4K	256	128	UART SPI	ADC 6×10	1×8 1×16	I
AT90LS8515	2,7...6,0	4	32	8K	512	512	UART SPI	–	2×8 1×16	I
AT90S8515	4,0...6,0	8	32	8K	512	512	UART SPI	–	2×8 1×16	I
AT90LS8535	2,7...6,0	4	32	8K	512	512	UART SPI	ADC 8×10	2×8 1×16	I
AT90S8535	4,0...6,0	8	32	8K	512	512	UART SPI	ADC 8×10	2×8 1×16	I
AT90S8534	1,8...6,0	4	32	8K	256	512	UART SPI	ADC 6×10	2×8 1×16	I

• таймери 8/16 — кількість і розрядність таймерів-лічильників;

• *ADC (channels)* — кількість каналів аналого-цифрового перетворення.

Широка номенклатура МК AVR дає змогу користувачу оптимізувати співвідношення продуктивність — енергоспоживання — ціна.

Висока продуктивність забезпечується:

• виконанням команд за один тактовий цикл;

Таблиця. 7.6. **Мікроконтролери AVR сімейства Mega**

Тип	Технічні характеристики									
	Напруга живлення, В	Тактова частота, МГц	I/O	Flash	EEPROM	SRAM	Інтерфейси	Аналогові входи	Таймери	ISP
AT-mega 8L	2,7...5,5	8	23	8K	512	1K	UART SPI	ADC 8×10	2×8 1×16	S
AT-mega 16L	2,7...5,5	8	32	16K	512	1K	UART SPI	ADC 8×10	2×8 1×16	S
AT-mega 16	4,5...5,5	16	32	16K	512	1K	UART SPI	ADC 8×10	2×8 1×16	S
AT-mega 32L	2,7...5,5	8	32	32K	1K	2K	UART SPI	ADC 8×10	2×8 1×16	S
AT-mega 32	4,5...5,5	16	32	32K	1K	2K	UART SPI	ADC 8×10	2×8 1×16	S
AT-mega 64L	2,7...5,5	8	53	64K	2K	4K	2×UART SPI	ADC 8×10	2×8 2×16	S
AT-mega 64	4,5...5,5	16	53	64K	2K	4K	2×UART SPI	ADC 8×10	2×8 2×16	S
AT-mega 103	4,0...5,5	6	48	128K	4K	4K	UART SPI	ADC 8×10	2×8 2×16	I
AT-mega 128L	2,7...5,5	8	53	128K	4K	4K	2×UART SPI	ADC 8×10	2×8 2×16	S
AT-mega 128	4,5...5,5	16	53	128K	4K	4K	2×UART SPI	ADC 8×10	2×8 2×16	S
AT-mega 161L	2,7...5,5	4	35	16K	512	1K	2×UART SPI	–	2×8 1×16	S
AT-mega 161	4,0...5,5	8	35	16K	512	1K	2×UART SPI	–	2×8 1×16	S
AT-mega 162L	2,7...5,5	8	35	16K	512	1K	2×UART SPI	–	2×8 1×16	S
AT-mega 162	4,5...5,5	16	35	16K	512	1K	2×UART SPI	–	2×8 1×16	S
AT-mega 163L	2,7...5,5	4	32	16K	512	1K	UART SPI	ADC 8×10	2×8 1×16	S
AT-mega 163	4,0...5,5	8	32	16K	512	1K	UART SPI	ADC 8×10	2×8 1×16	S

Продовж. табл. 7.6

Тип	Технічні характеристики									
	Напруга живлення, В	Тактова частота, МГц	I/O	Flash	EEPROM	SRAM	Інтерфейси	Аналогові входи	Таймери	ISP
AT-mega 169	2,7...3,6	4	53 4×25 LCD	16K	512	1K	UART SPI	ADC 8×10	2×8 1×16	S
AT-mega 8515L	2,7...5,5	8	35	8K	512	512	UART SPI	–	1×8 1×16	S
AT-mega 8515	4,5...5,5	16	35	8K	512	512	UART SPI	–	1×8 1×16	S
AT-mega 8535L	2,7...5,5	8	32	8K	512	512	UART SPI	ADC 8×10	1×8 1×16	S
AT-mega 8535	4,5...5,5	16	32	8K	512	512	UART SPI	ADC 8×10	1×8 1×16	S

• конвеєром команд, коли одночасно з виконанням поточної команди відбувається вибірка наступної;

• потужною системою команд єдиного 16-розрядного формату;

• вбудованими апаратними пристроями.

При цьому енергоспоживанню сприяє:

• CMOS технологія;

• статична робота — від покрокового режиму до максимальної тактової частоти.

Незначна вартість як на рівні вартості апаратного обладнання, так і на рівні вартості розробки й налагодження прикладних програм забезпечується:

• Flash-пам'яттю програм, що програмується без ...

• можливістю вибору мікроконтролера з достатньою і необхідною кількістю функцій і вбудованої периферії.

Нині співвідношення продуктивність — енергоспоживання — ціна для мікроконтролерів AVR є одним з кращих на світовому ринку 8-розрядних мікроконтролерів.

Контрольні запитання

1. Назвіть галузі застосування PIC-контролерів.
2. Якими факторами визначається швидкодія PIC-контролерів?
3. Назвіть основні блоки структурної схеми PIC-контролера.
4. Порівняйте характеристики PIC-контролерів серії PIC16Cxx і PIC17Cxx.
5. Які типи ПЗП, які використовують у PIC-контролерах?
6. Які периферійні пристрої застосовують у PIC-контролерах?
7. Назвіть склад периферії AVR-мікроконтролерів.
8. За якою архітектурою виконаний AVR-мікроконтролер та які особливості цієї архітектури?
9. Назвіть нові блоки структурної схеми AVR-мікроконтролера.
10. За скільки тактів виконується команда: а) в AVR-мікроконтролері; б) у PIC-контролері; в) у мікроконтролері MCS51?
11. Яке призначення сторожового WDT таймера?
12. Назвіть режими програмування Flash-пам'яті програм. Поясніть їхні переваги та недоліки.
13. Назвіть галузі застосування МК сімейств Tiny, Classic, Mega.
14. Які структурні особливості архітектур сімейств Tiny, Classic, Mega?
15. За рахунок чого досягається висока продуктивність AVR-мікроконтролерів? Порівняйте її з продуктивністю МК CISC- та PIC-контролерів.
16. Яка мінімальна частота роботи AVR-мікроконтролерів?
17. За рахунок чого досягається невеликий рівень енергоспоживання AVR-мікроконтролерів?
18. Які чинники сприяють зменшенню вартості AVR-мікроконтролерів?

Розділ 8

СИГНАЛЬНІ МІКРОПРОЦЕСОРИ

Сигнальні процесори належить до класу спеціалізованих МП (див. п. 1.1). Вони розроблені для розв'язання задач цифрового оброблення сигналів, прикладами якого є:

- фільтрація сигналу;
- згортка двох сигналів;
- обчислення значень кореляційної функції двох сигналів;
- обчислення автокореляційної функції;
- пряме-зворотне перетворення інтегралів Фур'є тощо.

Задачі цифрового оброблення розв'язуються в апаратурі зв'язку і передавання даних, засобах гідро- і радіолокації, медичному устаткуванні та робототехніці, керуванні двигунами, в автомобільній електроніці, телебаченні, вимірювальній техніці тощо.

Відмінною ознакою задач цифрового оброблення сигналів є потоковий характер оброблення великих обсягів даних у реальному режимі часу. Робота у реальному часі потребує підвищення швидкодії МП, а оброблення великих масивів даних — апаратних засобів інтенсивного обміну із зовнішніми пристроями.

Висока швидкодія сигнальних МП досягається завдяки:

- застосуванню модифікованої RISC-архітектури;
- проблемно-орієнтованій системі команд, наприклад, включенню в систему команд таких операцій, як множення з накопичуванням MAC ($C := A \times B + C$) із зазначеним у команді числом виконань у циклі та з правилом зміни індексів елементів масивів A і B;
- застосування способів скорочення тривалості командного циклу, наприклад конвеєризація команд;
- розміщенню операндів більшості команд у регістрах;
- використанню тіньових регістрів для збереження стану обчислень під час перемикання контексту;

• наявності апаратного множення, що дає змогу виконувати множення двох чисел за один командний такт;

• апаратній підтримці програмних циклів.

Сигнальні процесори різних компаній-виробників поділяють на два класи процесорів: на прості та дешеві мікропроцесори оброблення даних у форматі з фіксованою комою та на дорогі мікропроцесори, що апаратно підтримують операції над даними у форматі з плаваючою комою.

8.1. Сигнальні процесори оброблення даних у форматі з фіксованою комою

Перший сигнальний процесор TMS320C10, розроблений фірмою Texas Instruments у 1982 р., обробляв числа з фіксованою комою.

Структуру типового представника сімейства з фіксованою комою мікропроцесора TMS320XC5x зображено на рис. 8.1.

Процесор виконаний за гарвардською архітектурою, основою якої є розподіл шин доступу до вбудованої пам'яті програм і даних. Це дає змогу здійснити вибірку команди і даних в одному машинному циклі й забезпечити виконання більшості команд за один цикл.

Сигнальний процесор TMS320XC5x складається з центрального процесорного пристрою (CPU), вбудованої пам'яті програм і даних, багатофункціональних периферійних пристроїв, що у більшості випадків дають змогу позбутися додаткової зовнішньої апаратури.

Процесор містить шини *PDATA* — шина даних пам'яті програм; *PADDR* — шина адреси пам'яті програм; *DDATA* — шина даних пам'яті даних; *DADDR* — шина адреси пам'яті даних — для незалежного доступу до пам'яті програм і даних.

Центральний процесорний пристрій *CPU* містить: 32-розрядний арифметико-логічний пристрій *ALU*, який виконує більшість команд за один цикл; акумулятор (*ACC*), розділений на два сегменти по 16 розрядів (*ACCH* і *ACCL*); акумуляторний буфер *ACCB*; арифметичний пристрій допоміжних регістрів *ARAU*; регістровий файл *AR*0—*AR*7 і регістр *INDR*; незалежний логічний блок *PLU*; апаратний помножувач 16 × 16; регістри зсуву: масштабувальний регістр зсуву *SPL*, який здійснює зсув ліворуч від 0 до 16 розрядів, призначений для вирівнювання і перетворення даних, обраних з пам'яті, регістр зсуву *SFL* на виході помножувача;

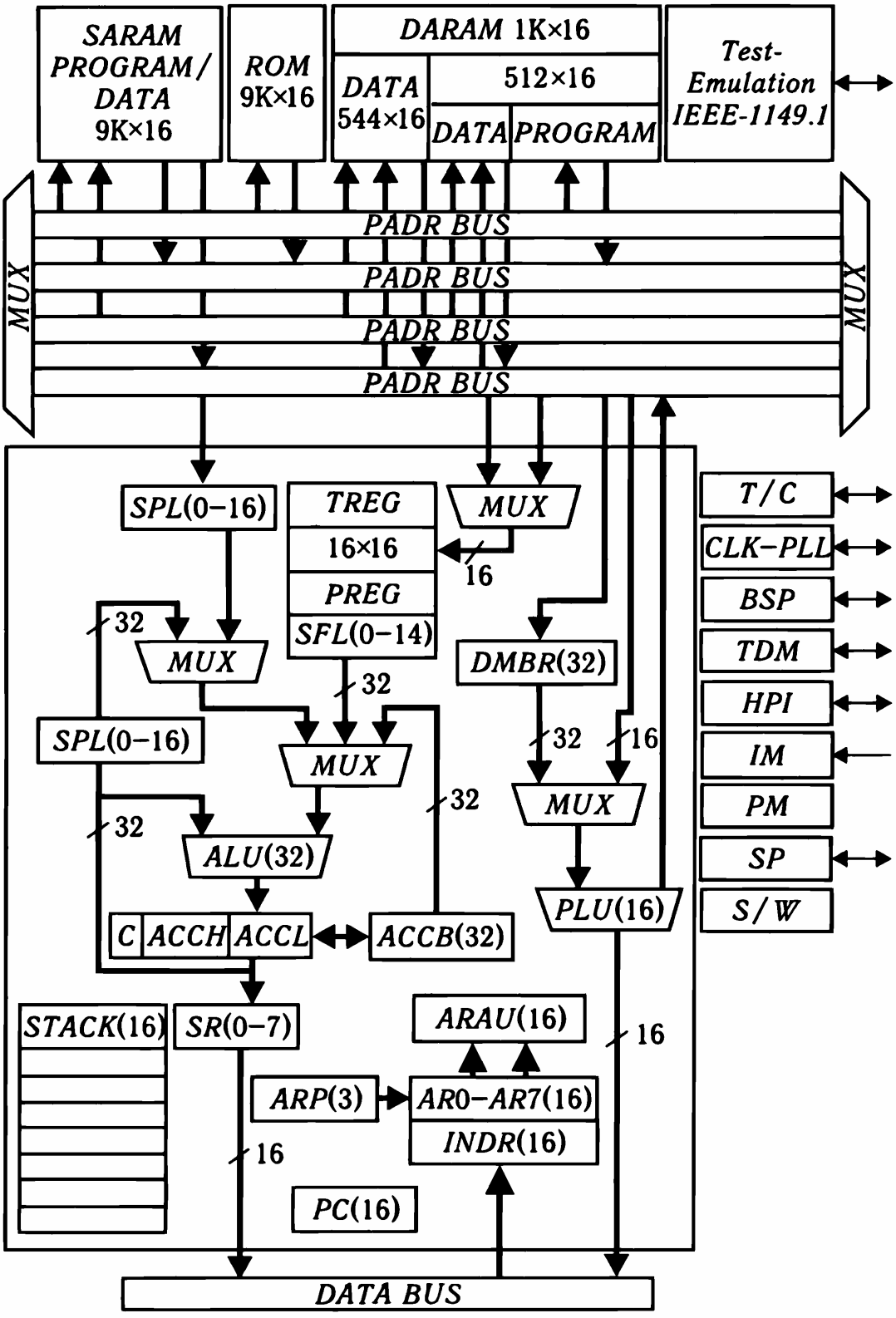


Рис. 8.1. Архітектура цифрового сигнального процесора TMS320XC5x

регістр зсуву *SR;* стек *STACK;* покажчик команд *PC;* мультиплексори *MUX.*

Арифметико-логічний пристрій ALU. На перший вхід *ALU* надходять дані одного з таких пристроїв:

масштабувального регістра зсуву *SPL*; регістра зсуву *SFL* на виході регістра помножувача *PREG*; акумуляторного буфера *ACCB*.

На другий вхід *ALU* дані завжди надходять з акумулятора *ACC*, а результат виконання операцій також надходить в *ACC*. Регістр зсуву *SR*, з'єднаний з виходом *ACC*, здійснює зсув ліворуч від 0 до 7 розрядів, що виконується в циклі пересилання даних з *ALU* на внутрішню шину даних.

Апаратний помножувач 16×16 виконує операції над числами зі знаком і без знака. Операнди надходять з пам'яті даних. Один з операндів може бути константою, наявною безпосередньо в команді. 16-Розрядний регістр *TREG* використовується для тимчасового збереження одного з операндів. У 32-розрядний регістр *PREG* завантажується результат множення.

Регістровий файл, що складається з восьми допоміжних (*AR*0—*AR*7) та індексного (*INDR*) регістрів, використовується для формування адреси для непрямої адресації. За потреби регістри *AR*0—*AR*7 можна використовувати для тимчасового збереження даних. Для адресації допоміжних регістрів є покажчик допоміжних регістрів (*ARP*). Регістри *AR*0—*AR*7 завантажуються з пам'яті даних, акумулятора *ACC* чи операндом, наявним у команді. Вміст *AR*0—*AR*7 може бути збережений у пам'яті чи використаний для обчислення в *ALU*.

Арифметичний пристрій ARAU разом з регістрами *AR*0—*AR*7 та *INDR* застосовують для генерації адреси. Звичайно регістри *AR*0—*AR*7 використовують для збереження адреси, а регістри *INDR* для зсуву. Прості арифметичні операції (додавання, віднімання, інкрементування, декрементування), виконувані в *ARAU* із вмістом *AR*0—*AR*7 і *INDR*, дають змогу реалізувати кілька видів непрямої адресації. Операції в *ARAU* виконуються одночасно з адресацією щодо поточної комірки пам'яті. Блок *ARAU* звільняє *ALU* від роботи з обчислення адрес.

Логічний блок PLU виконує операції незалежно від *ALU*. Результат операцій у *PLU* не впливає на біти стану *ALU*. Перший операнд надходить у *PLU* з пам'яті даних, другий — з пам'яті чи програм регістра маніпуляції бітами (*DBMR*). Спеціальні логічні команди, що виконуються лише *PLU*, дають змогу у 16-розрядному слові встановлювати та очищувати будь-яку кількість біт у довільній комбінації. Результат операцій у *PLU*

зберігається у тій самій комірці пам'яті, звідки був обраний перший операнд. Отже, логічні операції можна виконувати безпосередньо зі змістом будь-якої комірки пам'яті даних, у тому числі зі змістом перших 16 портів введення-виведення, що можуть адресуватися як пам'ять даних (адреси $50H-5FH$).

Пам'ять. МП TMS320XC5x передбачає роздільну адресацію до пам'яті програм, даних і портів введення-виведення. Розмір кожної області пам'яті 64 кілобайт 16-розрядних слів. Вбудована на кристалі пам'ять — *ROM, SARAM, DARAM* знаходиться у загальному адресному просторі пам'яті й може використовуватися як пам'ять програм чи даних.

Пам'ять програм ROM — масково-програмована пам'ять з можливістю захисту від зовнішнього доступу. У МП TMS320XC5x передбачено два режими роботи — мікропроцесорний і мікрокомп'ютерний. У мікрокомп'ютерному режимі *ROM* доступна, а в мікропроцесорному — закрита для доступу. Вибір режиму визначається рівнем напруги на вході *MP/MC* під час скидання. Після старту режим можна змінити програмно. Для зміни режиму роботи в регістрі стану процесорного режиму (*PMST*) передбачено біт керування (*MP/MC*).

Пам'ять даних чи ***програм-даних SARAM*** (*single access RAM*) передбачає виконання однієї операції — читання/запис у повному машинному циклі. *SARAM* складається з незалежних блоків по 2- чи по 1Кслів. Кожний блок допускає одну операцію читання-запис у машинному циклі. Тому в одному машинному циклі *CPU* може двічі звернутися до *SARAM*, але тільки тоді, якщо звернення відбувається до різних блоків. Цю особливість *SARAM* слід враховувати під час розподілу пам'яті та написанні програм. Інша особливість *SARAM* полягає в тому, що процесор чи інший зовнішній пристрій може звертатися безпосередньо до *SARAM* у режимі прямого доступу до пам'яті (*DMA*). Для ініціалізації доступу до *SARAM* процесор спочатку запитує доступ до зовнішньої пам'яті (встановлює сигнал *HOLD*). Після одержання сигналу підтвердження (*HOLDA*) процесор може запросити доступ до *SARAM* (встановлює сигнал *BR*). У цьому випадку *CPU* зупиняє всі поточні операції і підтверджує можливість доступу до *SARAM* (встановлює сигнал *IAQ*). Пам'ять *SARAM* можна використовувати для збереження лише даних, програм чи для спільного розміщення програм і даних. Конфігурація *SARAM* змінюється програмно за допомогою двох біт конфігурації — *OVLY* і *RAM*, що знаходяться в регістрі *PMST*. Можливість реконфігурації *SARAM* у процесі виконання програми дає змогу оперативно змінювати розподіл пам'яті МП TMS320XC5x.

Пам'ять даних DARAM (*dual access RAM*) передбачає виконання однієї операції читання та однієї операції запису в повному машинному циклі без конфліктів на внутрішній шині даних. *DARAM* складається з блоків B, B1, B2. Блоки B1 (32 слова) та B2 (512 слів) використовують лише як пам'ять даних, а блок B (512 слів) можна використовувати як пам'ять даних чи як пам'ять програм. Конфігурація блоку B змінюється програмно за допомогою біта конфігурації (*CNF*) у регістрі стану (*ST1*). Призначення блока B можна змінювати в процесі виконання програми. Передбачено можливість завантаження блока B програмним кодом із зовнішньої пам'яті з наступним виконанням.

Периферійні пристрої. ***Модуль переривання IM*** (*Interrupt module*) призначений для обслуговування зовнішніх, внутрішніх і програмних переривань.

До зовнішніх переривань належать два немаскованих переривання *RS*, *NM1* і п'ять маскованих *INT1*—*INT4*. Внутрішні переривання (*RINT, XINT, TRNT, TXNT*) генеруються послідовними портами або таймером (*TINT*). Програмні переривання викликаються командами *TRAP, INTR, NMI*.

Переривання встановлюють прапорець переривань у регістрі прапорців переривань (*IFR*) і можуть маскуватися у регістрі переривань (*IMR*).

Вектори переривань займають два 16-розрядних слова, необхідних для розміщення команд розгалуження. Після скидання МП вектори переривань розміщаються, починаючи з нульової адреси пам'яті програм. Під час звернення *CPU* до векторів переривань старші п'ять розрядів адреси формуються покажчиком векторів переривань (*IRTR*) у регістрі *PMST*.

Вбудований механізм захисту багатоциклових команд забезпечує вмикання механізму обслуговування переривань після завершення виконання багатоциклових команд. Дія механізму захисту поширюється також на команди, що стають багатоцикловими внаслідок їхнього повторення під час використання згідно з інструкцією *RPT* та на команди, що очікують завершення обміну з зовнішньою пам'яттю чи пам'яттю портами введення-висновку.

Блок керування енергоспоживанням PM (*Power management*). У МП TMS320XC5x передбачено три енергозберігаючих режими роботи (у тому числі й «сплячий» режим), в яких зниження струму споживання досягається відключенням *CPU* чи периферійних пристроїв. Перехід до енергозберігаючих режимів ініціюється активним сигналом

HOLD (режим захоплення зовнішньої шини) чи командами *IDLE, IDLE*2. Вихід з енергозберігаючих режимів здійснюється за сигналами зовнішніх переривань, що мають бути встановлені, принаймні, впродовж п'яти машинних тактів чи за внутрішніми перериваннями. В енергозберігаючих режимах зберігається стан усіх внутрішніх регістрів, що дає змогу без затримок продовжити роботу після виходу з цих режимів.

У режимі захоплення зовнішньої шини (*HOLD* = 0) продовжують працювати лише внутрішні ресурси МП TMS320XC5x. Зниження струму споживання відбувається за рахунок перемикання зовнішніх шин у високоімпедансний стан.

Інструкція *IDLE*2 зумовлює зупинку *CPU* і периферійних пристроїв («сплячий» режим), що значно знижує струм споживання.

Вбудований генератор CLK-PLL виробляє тактові синхросигнали для роботи *CPU* і периферійних пристроїв. Передбачено можливість підключення зовнішнього кварцового резонатора чи резонатора зовнішнього тактового генератора. Допускається режим роботи з множенням чи діленням частоти джерела тактового сигналу.

Таймер-лічильник T/C — це 16-розрядний лічильник, що працює на віднімання. Після досягнення нульового значення генерується переривання *TINT* і формується імпульс на виході *TOUT*. Тривалість імпульсу дорівнює періоду сигналу *CLKOUT*1. Керування таймера здійснюється програмно, і він може бути зупинений, перезапущений, відбутися його скидання чи заборона.

Інтерфейс Test-Emulation (*TE*) забезпечує можливість тестування мікросхем і підключення емулятора типу XDS510. Зв'язок з емулятором здійснюється по стандартному послідовному інтерфейсу IEEE1149.1 (*JTAG*).

Генератор тактів очікування S/W (*Waitstate Generator*) призначений для генерації і додавання в цикли обміну тактів очікування для збільшення тривалості циклів обміну з повільною зовнішньою пам'яттю чи портами введення-виведення. Використання генератора дає змогу обійтися без додаткової зовнішньої апаратури, що формує сигнал готовності *READY*. Генератор керується програмно. Кількість тактів очікування програмується окремо для пам'яті програм, даних, портів введення-виведення та областей адресного простору. Для керування генератором передбачено два регістри керування. Кількість тактів очікування може бути 0, 1,2, 3, 7.

Послідовний порт SP (*Serial port*) — стандартний послідовний порт, який дозволяє по шести лініях організувати повнодуплексний зв'язок між двома МП TMS320XC5x.

Для передавання даних в одному напрямі використовуються три лінії, по яких передаються: тактова частота, синхроімпульс, дані синхронно з тактовою частотою. Тактова частота й синхроімпульс формуються МП TMS320XC5x, але за потреби тактова частота та синхроімпульс можуть формуватися і зовнішніми пристроями. Можливі два режими передавання даних: пакетний, в якому синхроімпульс формується на початку кожного переданого слова; безперервний, в якому синхроімпульс формується лише на початку передавання. Вхідні та вихідні регістри зсувів буферизовані. Обмін по стандартному послідовному порту відбувається під керуванням *CPU*. Допускається 8- чи 16-розрядний формат передавання даних. Максимальна швидкість передавання даних залежить від тактової частоти МП TMS320XC5x. Якщо цикл 50 нс, максимальна швидкість передавання даних становить 5 Мбіт/с.

Послідовний порт з часовим поділом каналів TDM використовують для обміну даними між МП TMS320XC5x у мультипроцесорних системах. *TDM*-Порт працює у двох режимах, що переключаються програмно. Перший — режим стандартного послідовного порту, що був розглянутий вище, другий — режим часового поділу каналів, у якому для синхронізації передавання даних між процесорами МП TMS320XC5x кожні 128 тактів (*TCLK*) передається синхроімпульс (*TFRM*). По лінії даних (*TDAT*) передаються 16-розрядні дані, а по лінії адреси (*TADD*) — адреса. Керування роботою і контроль за станом *TDM*-порту здійснюється за допомогою шести регістрів.

Буферизований послідовний порт BSP складається з інтерфейсу послідовного порту (*SPI*), що є удосконаленою версією стандартного послідовного порту і блока автобуферизації (*ABU*). Незалежно від *CPU ABU* дає змогу виконувати обмін даними безпосередньо з вбудованою пам'яттю МП TMS320XC5x через спеціально виділену шину. Для буфера обміну даними використовують 2 кілобайти вбудованої пам'яті МП TMS320XC5x. Для адресації до пам'яті *ABU* має власний адресний регістр. Розмір і початкова адреса буфера запрограмовані. Допускається 8-, 10-, 12- або 16-розрядний формат передавання даних у пакетному чи безперервному режимі.

8-Розрядний паралельний порт HPI застосовують для обміну даними в мультипроцесорному режимі між *host*-процесором і МП TMS320XC5x. Host-інтерфейс *HPI* забезпечує можливість простої інтеграції процесора в мультипроцесорну систему. Обмін даними здійснюється через вбудовану буферну пам'ять розміром 2 кілослів по спеціальній

внутрішній шині, що дає змогу обмінюватися з пам'яттю без конфліктів з *CPU*. Буферна пам'ять — це *SARAM* пам'ять. Для керування *HPI* передбачено регістр керування-контролю (*HPIC*), що доступний host-процесору і *CPU*. Для адресації до буферної пам'яті з боку host-процесора використовують адресний регістр *HPIA*. *HPI* допускає два режими роботи: перший — режим *SAM*, в якому host-процесору і *CPU* дозволяється доступ до пам'яті, причому host-процесор має вищий пріоритет, ніж *CPU*; другий — режим *HOM*, в якому лише host-процесор має доступ до пам'яті. Для передавання через *HPI* одного байта даних треба п'ять машинних тактів. За тактової частоти 40 МГц максимальна швидкість передавання даних становить 64 Мбіт/с.

Блок початкового завантаження BL (*BOOT LOADER*) виконує пересилання програмного коду із зовнішніх джерел у вбудовану *RAM*-програму. Ініціалізація програми початкового завантаження відбувається після вмикання живлення лише в мікрокомп'ютерному режимі. Передбачено сім видів завантаження, що відрізняються способом і форматом передавання даних: через послідовний порт у 8- чи 16-розрядному форматі; через порти введення-виведення у 8- чи 16-розрядному форматі; із зовнішньої пам'яті у 8- чи 16-розрядному форматі; «гаряче завантаження». Вид завантаження визначається вмістом молодших восьми розрядів комірки глобальної пам'яті з адресою *FFFFH*, до якого МП TMS320XC5x звертається після вмикання живлення. Перед пересиланням програмного коду передається заголовок, що містить адреси початку розміщення програмного коду та довжину блока, що пересилається. Після завершення пересилання у *RAM*-програми МП TMS320XC5x стартує з адреси, що була зазначена у заголовку.

У МП TMS320XC5x доступ до зовнішньої пам'яті і портів введення-виведення здійснюється по шині адреси $AT-A15$ та по шині даних $D0-D15$ за допомогою керуючих сигналів *PS, DS, IS* (для вибору відповідно пам'яті програм, даних і портів введення-виведення), стробу *STRB*, сигналу напряму передавання в поточному циклі *R/W*, сигналу читання *RD* і сигналу запису *WE*. Максимальна продуктивність забезпечується під час обміну з високошвидкісною зовнішньою пам'яттю, що дає змогу працювати без тактів очікування. Можливе підключення повільної і більш дешевої пам'яті. У цьому разі в цикли читання-запис МП TMS320XC5x слід додавати такти очікування, які генеруються вбудованим генератором тактів очікування, чи формувати зовнішній сигнал *READY* — готовності зовнішньої пам'яті або портів введення-виведен-

ня. Під час організації обміну з зовнішньою пам'яттю треба також враховувати, що цикли читання мають тривалість одного машинного такту, тоді як тривалість циклів запис становить два машинних такти, а якщо запис відбувається безпосередньо за читанням, то навіть три такти. У верхніх адресах пам'яті даних може розміщуватися зовнішня глобальна пам'ять даних розміром від 256 до 32 кілослів. Адреси з $00H$ до $5FH$ пам'яті даних відведені під внутрішні регістри. Перші 16 портів введення-виведення розміщені у пам'яті даних. Тому звертання до цих портів можливе не лише за допомогою команд *IN* і *OUT*, а й за допомогою звичайних команд звернення до пам'яті (завдовжки 1 слово), що сприяє зменшенню розмірів програмного коду і збільшенню швидкості обчислення.

8.2. Сигнальні процесори оброблення даних у форматі з плаваючою комою

Використання сигнальних процесорів для оброблення даних у форматі з плаваючою комою зумовлюється рядом задач (інтегральні перетворення, алгоритми компресії, декомпресії, адаптивної фільтрації), які вимагають високої точності подання даних у широкому динамічному діапазоні. Робота з даними у форматі з плаваючою комою спрощує і прискорює їх оброблення, підвищує надійність програми, оскільки не потребує виконання операцій округлення і нормалізації даних, відстеження ситуацій втрати значимості та переповнення. Однак апаратні й вартісні затрати таких МП значно більші, ніж процесорів оброблення даних у форматі з фіксованою комою.

Першим представником класу процесорів з плаваючою комою був МП TMS320C30, структурну схему якого зображено на рис. 8.2.

Процесор має 32-розрядну шину команд і даних та 24-розрядну шину адреси, містить 2 блоки ОЗП по 1К 32-розрядних слів, 32-розрядний блок множення з плаваючою комою, кеш-пам'ять команд ємністю 64 32-розрядних слів, 8 регістрів для операцій з підвищеною точністю, 2 генератори адреси і регістровий файл, реалізує різні способи адресації. 40-Розрядне АЛП процесора працює як з цілими числами, так і з числами у форматі з плаваючою комою. Вбудований контролер ПДП дає змогу сполучати в часі обчислення та обмін даними з пам'яттю. Наявність у МП TMS320C30 мультипроцесорного інтерфейсу, двох зовнішніх інтерфейсних портів, двох

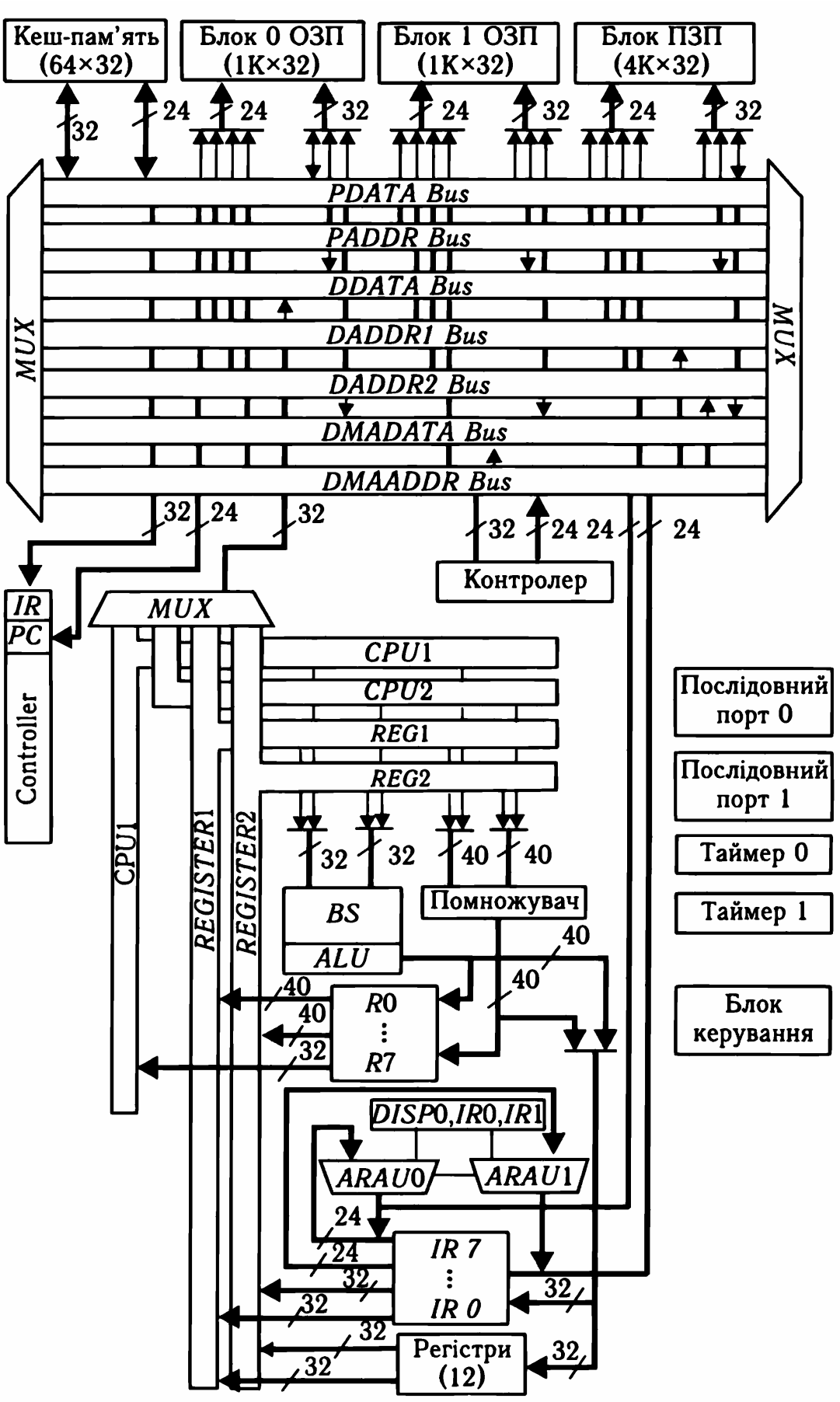


Рис. 8.2. Архітектура цифрових сигнальних процесорів TMS320C3x

послідовних портів, розширеної системи переривань спрощує конструювання систем на його основі.

Усі операції в процесорі виконуються за один такт. Процесор може паралельно виконувати в одному такті операцію множення та арифметико-логічну операцію з числами у форматі з фіксованою чи плаваючою комою. Процесор має гнучку систему команд та підтримку мови високого рівня *C*.

Процесор містить кілька шин:

- *PDATA* — шина даних пам'яті програм;
- *PADDR* — шина адреси пам'яті програм;
- *DDATA* — шина даних пам'яті даних;
- *DADDR1* — шина адреси пам'яті даних *Block0;*
- *DADDR2* — шина адреси пам'яті даних *Block1;*
- *DMADATA* — шина даних пам'яті в режимі ПДП;
- *DMAADDR* — шина адреси пам'яті в режимі ПДП,

які використовують для незалежного доступу до вбудованої пам'яті програм і даних. 32-Розрядні шини даних і 24-розрядні шини адреси забезпечують можливість незалежного вибору операндів з різних джерел.

Центральний процесорний пристрій *CPU* містить внутрішні шини даних *CPU*1 та *CPU*2 і регістрові шини *REG*1 і *REG*2. Шини *CPU*1 і *CPU*2 дають можливість процесору передавати два операнди з пам'яті даних та регістрового файла за один машинний цикл. Регістрові шини *REG*1 і *REG*2 можуть за один машинний цикл передавати два слова даних з регістрового файла в помножувач і арифметико-логічний пристрій *ALU*. Тому чотири незалежні внутрішні шини *CPU* забезпечують паралельне виконання команд в *ALU* і помножувачі в одному циклі. Помножувач і *ALU*, що оперує над цілими числами у форматі з плаваючою комою, забезпечують високу продуктивність і точність обчислень.

До складу *CPU* входять також 32-розрядний регістр зсуву *BS* (*Barrel Shifter)*, регістровий файл (28 регістрів), два арифметичних пристрої допоміжних регістрів *ARAU*0 і *ARAU*1, що разом з вісьмома допоміжними регістрами *AR*0—*AR*7 забезпечують гнучкі режими непрямої адресації. Крім перелічених пристроїв *CPU* містить також конвеєр, механізм контролю за перериваннями, лічильники переривань і повторень, механізм умовних переходів.

Арифметико-логічний пристрій ALU виконує за один цикл арифметичні операції з 32-розрядними цілими числами та 40-розрядними числами у форматі з плаваючою комою, а також логічні операції з 32-розрядними операндами. 32-Розрядний кільцевий регістр зсуву *BS* дає змогу за один цикл виконувати зсув праворуч чи ліворуч на 1...32 розрядів.

Помножувач виконує множення 24-розрядних цілих чисел і 32-розрядних чисел у форматі з плаваючою комою. У першому випадку результатом є 32-розрядне число, а в другому — 40-розрядне число, подане у форматі з плаваючою комою.

Регістровий файл CPU складається з 28 регістрів. Частина регістрів доступна помножувачу та *ALU* і може використовуватися як регістри загального призначення. Вісім 40-розрядних регістрів розширеної точності *R*0—*R*7 підтримують операції з цілими числами та числами у форматі з плаваючою комою. Вісім 32-розрядних допоміжних регістрів *AR*0—*AR*7 доступні *ALU* і пристроям *ARAU*. Регістри *AR*0—*AR*7 можна використовувати для організації циклічних лічильників. Два 32-розрядних індексних регістри *IR*0, *IR*1 використовуються пристроями *ARAU* для індексації адреси. Крім перелічених регістрів у регістровий файл входять покажчик системного стеку *SP* та регістри стану *ST*, прапорців переривань *IF*, масок переривань *IE*, адреси початку повторень *RS* і адреси кінця повторень *RE*, призначені для керування процесом обчислення.

Арифметичні пристрої ARAU0 і ***ARAU1*** можуть упродовж одного циклу генерувати дві адреси на шинах *DADDR*1 і *DADDR*2. *ARAU* підтримують адресацію зі зсувом, з індексацій (за допомогою регістрів *IR*0 і *IR*1) і циклічну. Виконання операцій в *ARAU* здійснюється одночасно з операціями в *ALU* і помножувачі.

CPU підтримує роботу з трьома форматами подання чисел: цілі числа зі знаком і без знака, числа у форматі з плаваючою комою. Цілі числа можна подати в короткому форматі (діапазон подання чисел знаходиться в межах $-2^{15} < n < 2^{15} - 1$) та форматі одинарної точності (діапазон подання чисел знаходиться в межах $-2^{31} < n < 2^{31} - 1$). Числа з плаваючою комою можна подати в короткому і розширеному форматах та форматі одинарної точності. Для подання чисел з плаваючою комою у розширеному форматі треба 40 розрядів, 8 з яких призначені для значення показника степеня, а 32 — для значення мантиси. Для розширеного формату подання чисел максимальне додатне число становить $3{,}4028236683 \times 10^{38}$, а мінімальне від'ємне число — $-3{,}4028236691 \cdot 10^{38}$. Операції над числами з плаваючою комою забезпечують точні обчислення і дають змогу ліквідувати проблеми переповнення і вирівнювання операндів, що виникають у цілочисловій арифметиці. Для перетворення форматів чисел (тобто з цілочислового у формат з плаваючою комою і навпаки), округлення і нормалізації передбачені спеціальні команди.

Незалежні шини доступу до вбудованої пам'яті, внутрішні шини *CPU*, а також можливість одночасної генерації пари адрес дають змогу реалізувати гнучкі режими адресації, що забезпечують вибірку даних з пам'яті, регістрового чи файла команд. У *CPU* використовуються такі типи адресації: регістрова, пряма, непряма (27 видів), безпосередня, *PC*-відносна (*PC — program counter).* Перелічені типи адресації використовують у різних режимах адресації — триопераидному, паралельному, циклічному і реверсивному.

Команди з триопераидним режимом адресації виконують вибірку двох операндів, операцію з обраними операндами і запис результату операцій у регістр, причому ці дії виконуються в одному циклі.

Команди паралельних операцій здійснюють паралельне завантаження регістрів, паралельні арифметико-логічні операції та операції завантаження. Всього передбачено 28 подібних команд. Наприклад, одна з них (*MPYF3//ADDF3*) дає змогу перемножити і скласти дві пари чисел у форматі з плаваючою комою, інша (*MPYI3//SUBI3*) — перемножити і відняти дві пари цілих чисел.

Циклічна адресація застосовується для реалізації багатьох алгоритмів цифрового оброблення сигналів — згортка, кореляція тощо.

Реверсивна адресація знаходить широке застосування під час реалізації алгоритмів швидкого перетворення інтегралів Фур'є.

CPU має механізми, що дають змогу керувати процесом виконання програми. Це лічильник повторень, механізм переходів (стандартних і затриманих) та контролер переривань.

Лічильник повторень використовується для виконання команд з префіксами: *RPTB* — повторення блока команд і *RPTS* — повторення однієї команди, що сприяє спрощенню організації циклів. Подібні цикли застосовують під час реалізації багатьох алгоритмів цифрового оброблення сигналів, у яких є повторні групи команд, що займають велику частину часу реалізації алгоритму. Використання повторень дає змогу скоротити тривалість реалізації подібних алгоритмів.

Контролер переривань обробляє зовнішні, внутрішні та програмні переривання. До зовнішніх переривань належать переривання скидання *RESET* та переривання, що надходять на виводи ВІС *INT*0—*INT*3. Внутрішні переривання формуються послідовними портами *XINT*0, *RINT*0, *XINT*1, *RINT*, таймерами *TINT*0 та *TINT*1 і контролером *DMA DINT*. Програмні переривання викликаються командою *TRAP*. Вектори переривань займають адреси від 00*H* до 3*FH*.

Переривання можуть обслуговуватися як *CPU*, так і контролером *DMA*. Контролер *DMA* використовує переривання для синхронізації операцій пересилання даних. Переривання встановлюють прапорець переривання у регістрі прапорців переривань *IF* і можуть маскуватися в регістрі *IE* маскування переривань.

CPU мікропроцесора TMS320C31 передбачає два режими роботи зі зменшеним споживанням струму, що задаються командами *IDLE2* і *LOPOWER*. Зниження струму споживання в першому випадку досягається зупинкою *CPU*, а в другому — зменшенням у 16 разів тактової частоти мікропроцесора. Вихід з цих режимів здійснюється за сигналами зовнішніх переривань. Споживання струму МП TMS320C31 з тактовою частотою 50 МГц приблизно становить 250 мА. У режимі *IDLE2* струм споживання знижується до 50 мА .

Пам'ять. Загальна ємність пам'яті TMS320C31 становить 16 М × 32 слів. У ньому розміщаються області пам'яті програм, даних і пристроїв введення-виведення. Команди, дані та таблиці можуть зберігатися у будь-якому місці загальної пам'яті.

Вбудована пам'ять складається з 4 Кбайт *ROM* (лише для МП TMS320C30), 2 Кбайт *RAM* (2 блоки по 1 Кбайт) і 64 слів кеш-пам'яті програм. У мікропроцесорах TMS320C31 і TMS320C32 замість *ROM* вбудований блок початкового завантаження (*Boot Loader*), причому TMS320C32 має всього 512 слів *RAM*-пам'яті (2 блоки по 256 слів). *ROM* і *RAM* пам'ять підтримують подвійне звернення за один цикл. Незалежні шини даних *PDATA, DDATA, DMADATA* дають змогу в одному циклі здійснити вибірку команд, читання-запис даних та операцію прямого доступу до пам'яті *DMA*.

МП TMS320C3x може працювати у двох режимах — мікропроцесорному і мікрокомп'ютерному. Режим роботи визначається станом входу *MC/MP*.У мікрокомп'ютерному режимі *ROM* розміщується з адреси 000*H* до *FFFH*. За адресами з 00*H* до 3*FH* розташовані вектори переривань. У мікропроцесорному режимі вбудована *ROM* маскується й у зазначених адресах розміщується зовнішня пам'ять.

Кеш-пам'ять програм призначена для збереження часто повторюваних наборів команд, що дає змогу зменшити кількість звернень до зовнішньої пам'яті й використовувати повільну і значно дешевшу зовнішню пам'ять. Зовнішні шини МП TMS320C3x звільняються у такий спосіб для операцій *DMA* чи введення-виведення даних. Вибірка команд з вбудованої пам'яті (*ROM* чи *RAM*) не модифікує кеш-пам'ять. Читання-запис даних у кеш-пам'яті не виконується. Перед-

бачено чотири режими роботи кеш-пам'яті, що встановлюються програмно.

Зовнішня пам'ять. Обмін із зовнішньою пам'яттю і пристроями введення-виведення здійснюється через основну та додаткову шини (див. рис. 8.2).

Основна шина має шини даних *D*31 — *D*0 та адреси *A*23 — *A*0 і керуючі сигнали — строб звертання до пам'яті *STRB*, сигнали читання-запис *R/W*, готовності *RDY*, захоплення основної шини *HOLD* і підтвердження захоплення *HOLDA*. До складу додаткової шини входять шини даних *XD*31 — *XD*0 та адреси *XA*12 — *XA*0 і керуючі сигнали — строби звертання до пам'яті *MSTRB* та звертання до пристроїв введення-виведення *IOSTRB*, сигнали читання-запис *XR/W* і готовності *XRDY*. Основна і додаткова шини мають відповідні регістри керування.

Для забезпечення максимальної швидкодії процесора, тобто без тактів очікування, будь-який пристрій, що підключається до МП, повинен мати тривалість вибірки не більш як 25 нс (за тактової частоти МП TMS320C3x — 33 Мгц). У разі використання повільної і більш дешевої пам'яті значно збільшується гнучкість і знижується вартість системи, але в циклі звертання до пам'яті чи до пристроїв введення-виведення слід додавати такти очікування МП. TMS320C3x дає змогу формувати такти очікування як на основній, так і на додатковій шинах. При цьому обидві шини мають незалежну логіку керування готовності пам'яті чи пристроїв введення-виведення. Такти очікування можуть формуватися внутрішнім генератором тактів очікування чи зовнішнім сигналом готовності. Внутрішній генератор тактів очікування може формувати від 0 до 7 тактів (керується програмно), причому кількість тактів однакова для всіх зовнішніх циклів незалежно від використовуваної адреси. Крім того, для формування внутрішнього сигналу готовності в МП TMS320C3x передбачена можливість об'єднання сигналу від внутрішнього генератора тактів очікування і зовнішнього сигналу *RDY* за схемою ЛОГІЧНОГО І або ЛОГІЧНОГО АБО, що дає змогу більш гнучкіше використовувати зовнішні ресурси. Вибір режиму формування внутрішнього сигналу готовності здійснюється програмно.

У МП TMS320C3x передбачено механізм програмного перемикання банків пам'яті, що значно полегшує розробку систем, в яких використовується велика ємність пам'яті. На тривалість перемикання банків сигнал *STRB* стає неактивним на один такт. За такого способу усувається необхідність додавання тактів очікування, формованих сигналом *RDY* під час перетинання меж банків пам'яті. Розмір банку пам'яті

встановлюється програмно. Механізм перемикання банків пам'яті передбачений лише для основної шини.

Обмін на зовнішніх шинах має певні обмеження за швидкодією, оскільки тривалість циклу *читання* зовнішньої пам'яті становить один такт за відсутності тактів очікування, тоді як тривалість циклу *запис* становить два чи навіть три такти, якщо запис відбувається безпосередньо за читанням. Тривалість циклів *читання-запис* пристроїв введення-виведення (активний сигнал *IOSTRB*) на додатковій шині за відсутності тактів очікування триває два такти.

Контролер DMA виконує пересилання даних усередині адресного простору пам'яті та пристроїв введення-виведення без участі *CPU*, що дає змогу вести обмін з повільними пристроями пам'яті й пристроями введення-виведення (кодеки, послідовні порти, АЦП тощо), не зменшуючи продуктивності *CPU*. Пересилання даних за керуванням контролера *DMA* складаються з операцій читання і запис. Пересилання даних завершується лише тоді, коли повністю виконана як операція читання, так і операція запис. Пересилання даних може бути синхронізоване за «джерелом», «приймачем» чи «джерелом і приймачем», тобто окремі операції (читання або запис чи обидві разом) не виконуються доти, поки не виникне відповідне переривання. Контролер *DMA* і швидкісні зовнішні шини (основна і додаткова) є могутнім засобом для побудови високошвидкісних систем оброблення даних. Робота контролера *DMA* керується чотирма регістрами: загального керування, адресами джерела і приймача та регістром лічильника пересилань. 24-Розрядні регістри адреси джерела і приймача можуть інкрементуватися чи декрементуватися незалежно один від одного. 24-Розрядний регістр лічильника пересилань, керований 24-розрядним лічильником, використовується для завдання розміру блока, що пересилається. Крім перелічених регістрів у керуванні контролером *DMA* використовується регістр *IE* маскування переривань.

Таймер T/C — 32-розрядний таймер-лічильник подій із зовнішнім чи внутрішнім тактуванням. Таймер має зовнішній вихід, що використовується як вихід сигналу таймера, або як вхід для тактових імпульсів. Таймер можна використовувати для генерації сигналів, переданих зовнішнім пристроям (наприклад, запуск АЦП), чи для підрахунку числа зовнішніх імпульсів з наступною генерацією переривань у *CPU* після досягнення заданих показань лічильника. Для керування роботою таймера передбачені регістри керування. 32-Розрядний регістр лічильника містить поточне значення лічильника. Зміст лічильника може інкрементуватися за наростаю-

чим чи спадним фронтом тактового сигналу. 32-Розрядний регістр загального керування таймера призначений для завдання режимів роботи таймера і керування.

Послідовні порти SP *(Serial Port)* дають змогу організувати повнодуплексний зв'язок між двома МП TMS320C3x по шести лініях. Послідовні порти цілком незалежні й ідентичні щодо керування. Для передавання даних в одному напрямі використовуються три лінії, по яких передаються: тактова частота, дані, синхроімпульси. Допускаються 8-, 16-, 24- і 32-розрядний формат передавання даних, причому воно може здійснюватися як у фіксованому, так і в змінному форматі. Можливі два режими передавання даних — безперервний і пакетний. Тактова частота і синхроімпульси можуть бути як внутрішніми, так і формуватися зовнішніми пристроями. Вхідні та вихідні зсувні регістри буферизовані. Обмін даними через послідовний порт відбувається за керуванням *CPU*.

Блок початкового завантаження *(Boot Loader)* виконує пересилання програмного коду із зовнішньої пам'яті *(EPROM)* чи через послідовний порт за певною адресою. Завантаження із зовнішньої пам'яті може здійснюватися у 8-, 16- чи 32-розрядному форматі. Завантаження через послідовний порт виконується в 32-розрядному форматі в пакетному режимі. Під час завантаження через послідовний порт синхроімпульси і тактова частота формуються зовнішнім пристроєм. Програма початкового завантаження ініціюється за сигналами переривань *INT*3—*INT*0 лише у мікрокомп'ютерному режимі. Після старту перевіряється стан прапорців переривань у регістрі *IF* і визначається вид завантаження. Кожному прапорцю переривань (*INT*3, *INT*2, *INT*1, *INT*0) відповідає визначений вид завантаження.

Блокові програмного коду, що пересилається, передує заголовок, що містить крім коду формату передачі (у разі завантаження із зовнішньої пам'яті) розмір блока, що пересилається, і початкова адреса розміщення програмного коду. Після завершення пересилання програмного коду починається виконання програми з початкової адреси розміщення блока.

Інші BIC-представники сімейства МП TMS320C3x переважно відрізняються кількістю послідовних портів і каналів прямого доступу до пам'яті *DMA*.

Основне застосування мікропроцесорів сімейства TMS320C3xx — цифрове аудіо, 3-*D* графіка, відеоконференцзв'язок, промислові роботи, копіювально-розмножувальна техніка, телекомунікаційні системи.

8.3. Технічні характеристики сигнальних процесорів

Сигнальні процесори фірми Texas Instruments. Наступними за розглянутими в п. 8.1. і 8.2 сімействами сигнальних процесорів фірми Texas Instruments стали процесори оброблення даних з плаваючою комою TMS320C4x. Вони сумісні за системою команд із МП TMS320C3x (див. п. 8.2), однак мають більшу продуктивність і кращі комунікаційні можливості.

У сімейство сигнальних процесорів TMS320C4x входять такі процесори: TMS320C40, TMS320C44, TMS320LC40.

Процесор TMS320C40 має продуктивність 30 MIPS/60MFLOPS і максимальну пропускну здатність підсистеми введення-виведення 384 Мбайт/с. Він містить на кристалі 6 високошвидкісних (20 Мбайт/с) комунікаційних портів і 6 каналів DMA, 2Кслів пам'яті, 128 слів програмної кеш-пам'яті та блок початкового завантаження. Дві зовнішні шини забезпечують 4Гслів загального адресного простору.

Процесор TMS320C44 — значно дешевший варіант попереднього представника сімейства, що має 4 комунікаційних порти і загальний адресний простір 32Мслів. Однак значення показників продуктивності й пропускної здатності процесора ті самі, що й у попереднього представника сімейства.

Процесор TMS320LC40 — аналог МП TMS320C40, що відрізняється низьким енергоспоживанням, підвищеною продуктивністю (40 MIPS/80 MFLOPS) і більшою пропускною здатністю (488 Мбайт/с).

Подальшим розвитком сімейства цифрових процесорів компанії Texas Instruments для оброблення сигналів є процесор принципово нової архітектури — TMS320C80, випущений наприкінці 1994 р. Процесор призначений для високопродуктивного цифрового оброблення сигналу в самих широких галузях науки і техніки. Інша назва процесора — MVP (*Multimedia Video Processor*) — характеризує його високу ефективність для оброблення зображень, застосування у 2- і 3-вимірній графіці, у системах віртуальної реальності, компресії та декомпресії відео- і аудіоданих, а також для оброблення інформації у зв'язку тощо.

Процесор TMS320C80 поєднує в одній мікросхемі п'ять повнофункціональних процесорів, чотири з них — поліпшені цифрові процесори оброблення сигналів (*Advanced Digital Signal Processor),* архітектура яких орієнтована на реалізацію алгоритмів цифрового оброблення сигналів. Кожний з *ADSP* дає змогу виконати за один командний такт кілька *RISC*-подібних операцій. П'ятий процесор, головний (*Master Processor* —

MP), — це 32-розрядний *RISC*-процесор з високопродуктивним обчислювачем з плаваючою крапкою. Додатково до розглянутих вище процесорів у цьому МП розміщені:

• контролер обміну (*Transfer Controller (TC)*) — інтелектуальний контролер ПДП, що підтримує інтерфейс із *DRAM* і *SRAM*;

• відеоконтролер (*Video Controller — VC*);

• система контролю і налагодження — порт *JTAG* (*IEE* 1149.1)

• 50 Кбайт *SRAM*.

Продуктивність процесора TMS320C80 досягає 2 млрд *RISC*-подібних команд за секунду. Пропускна здатність шини досягає 2,4 Гбайт/с у потоці даних та 1,8 Гбайт/с — у потоці команд.

Процесор TMS320C80 має такі технічні характеристики:

• тактова частота 40 чи 50 Мгц.

• 50 Кбайт вбудованої *SRAM*;

• 64-розрядний контролер обміну з динамічним конфігуруванням шини на обмін 64-, 32-, 16- і 8-розрядними словами;

• режим ПДП до 64-розрядної пам'яті *SRAM* та *DRAM*;

• 4 зовнішні переривання;

• вбудовані засоби внутрішньосхемної емуляції;

• напруга живлення 3,3 В;

• близько 4 000 000 транзисторів на кристалі;

• 0,5/0,6 КМОП-технологія;

• 305-контактний корпус PGA.

Нове сімейство DSP-процесорів компанії Texas Instruments TMS320C62x матиме процесори як з фіксованою, так і з плаваючою крапкою. Перший представник цього сімейства — процесор TMS320C6201, що оперує з даними у форматі з фіксованою крапкою.

Побудований відповідно до розробленої компанії Texas Instruments архітектурою VelociTI процесор TMS320C62xx — перший із сигнальних VLIW — процесорів, що використовує для підвищення продуктивності паралелізм рівня команд. Традиційна VLIW-архітектура передбачає наявність кількох функціональних пристроїв, що працюють паралельно та виконують за один такт кілька команд. Архітектура VelociTI дає змогу забезпечити кращу ефективність за рахунок ослаблення обмежень на порядок і спосіб виконання команд. Основою МП TMS320C6201 є VelociTI VLIW — процесор з 8-функціональними модулями, включаючи 2 помножувачі та 6 АЛП. Модулі взаємодіють через два регістрових файли, кожний з яких містить по шістнадцять 32-розрядних регістри. ЦП може виконувати до 8 команд за один такт.

У процесорі використовується упакування команд, що скорочує розміри коду і тривалість вибірки команд. 256-розрядна шина пам'яті програм дає змогу вибирати за один такт вісім 32-розрядних команд. Усі інструкції містять умови їхнього виконання, що дає змогу скоротити витрати продуктивності процесора на виконання переходів і збільшити ступінь паралелізму оброблення даних.

Процесор може оперувати з 8-, 16- та 32-розрядними даними. Для додатків, що потребують високої точності обчислень, передбачена можливість обчислень з 40-розрядними операндами. Для результатів усіх основних арифметичних операцій виконується округлення і нормалізація. У процесорі реалізовані операції над бітовими полями, такі як «виділити» (*extract*), «встановити» (*set*), «очистити» (*clear*), «підрахунок бітів» (*bit counting*).

Процес оброблення *VLIW* починається з вибірки з пам'яті команд 256-бітового пакета. Команди зв'язуються для спільного виконання у виконуваний пакет (до 8 команд) за значенням молодшого біта команди (*LSB*).

Пристрій вибірки-декодування-диспетчеризації команд може направляти до функціональних модулів до восьми 32-розрядних команд за один такт для кожного із способів оброблення — *A* і *B*.

Процесор реагує на 14 переривань, що відповідають сигналу Reset, немаскованому перериванню (*NMI*) та перериванням з номерами 4—15.

Внутрішньокристальна пам'ять поділена на пам'ять даних та пам'ять програм (по 64 Кбайт). Процесор TMS320C6201 має два 32-розрядних порти для звернення до пам'яті даних та один 256-розрядний порт для звернення до пам'яті програм для вибірки інструкцій. Він має по 64 Кбайт пам'яті даних і програм на кристалі. У процесорі використано розшарування пам'яті даних (чотири 16-розрядних банки) для підвищення швидкості вибірки за рахунок одночасного звертання до різних банків пам'яті.

Додатково процесори сімейства TMS320C62xx можуть містити на кристалі інтерфейс зовнішньої пам'яті (*EMIF*), контролер ПДП (*DMA*), інтерфейс host-порта (*HPI*), засобу енергозбереження, розширені буферизовані послідовні порти, 32-розрядні таймери

До числа нових галузей застосування мікропроцесорів сімейства TMS320C62xx компанія Texas Instruments належать:

- універсальний безпровідний зв'язок;
- медична діагностика;
- телефонія;

• персональні засоби інформаційного забезпечення і захисту.

Крім того, можливе використання мікропроцесорів TMS320C62x в існуючих прикладних системах для збільшення їхньої продуктивності. До таких систем належать:

• базові станції мобільного зв'язку;

• модемні пули і сервери;

• кабельні модеми;

• багатоканальні телефонні платформи, включаючи центральні офісні комутатори, і системи мовного передавання інформації;

• мультимедійні системи.

Сигнальні процесори фірми Motorola. Сигнальні мікропроцесори компанії Motorola поділяють на сімейства 16- та 24-розрядних мікропроцесорів з фіксованою комою *DSP* (*Digital Signal Processor* — цифровий сигнальний процесор) — DSP560xx, DSP561xx, DSP563xx, DSP566xx, DSP568xx і мікропроцесори з плаваючою комою DSP960xx.

16-Розрядні мікропроцесори з фіксованою комою DSP56156 і DSP56166 насамперед призначені для використання в системах оброблення аудіосигналів і телекомунікацій.

Процесори цього сімейства мають такі технічні характеристики:

• продуктивність до 30 *MIPS* на частоті 60 Мгц;

• однотактовий паралельний помножувач (16 × 16 розрядів) з накопиченням *(MAC);*

• два 40-розрядних акумулятори з байтом розширення;

• гнучка система адресації;

• апаратна реалізація вкладених циклів, включаючи нескінченні цикли;

• три 16-бітові внутрішні шини даних і три 16-бітові внутрішні шини адреси;

• програмована тривалість доступу до зовнішньої шини;

• інтерфейс з відображенням у пам'ять периферійних пристроїв;

• внутрішній блок емуляції та налагодження (*OnCE*);

• низьке споживання енергії і засоби енергозбереження;

• 2 Кслів внутрішньокристальної пам'яті.

24-Розрядні мікропроцесори з фіксованою комою DSP56000/DSP56001 орієнтовані на максимізацію пропускної здатності в додатках *DSP* з інтенсивним обміном даними. Мікропроцесори працюють на частотах до 33 МГц і забезпечують продуктивність близько 16 *MIPS,* що дає змогу виконувати швидке перетворення інтегралів Фур'є по 1024 відліках за 3,23 мс. Відмінність між цими процесорами полягає у типі їх внутрішньої пам'яті. З метою мінімізації вартості

прикладних систем мікропроцесор DSP56000 орієнтований на роботу з використанням програми, збереженої в ППЗУ (*ROM*) ємністю 3,75 Кслів. Існує також варіант процесора DSP56000, що має захист від несанкціонованого доступу до програми, збереженої у внутрішній пам'яті.

Наступна лінія мікропроцесорів DSP56300 містить процесорне ядро нового типу (*NDE — New DSP Engine*), в якому завдяки конвеєризації забезпечується виконання команди за один такт, що підвищує швидкодію порівняно з ядром DSP560xx вдвічі.

32-Розрядний мікропроцесор з плаваючою комою сімейства DSP96002 містить 1024 слова пам'яті, рівномірно поділеної між пам'яттю даних *X* і *Y*, 1024 слова програмної пам'яті, два ППЗУ даних, двоканальні контролери ПДП, підсистему початкового завантаження програми і вбудовані засоби налагодження та емуляції.

Процесори сімейства мають такі технічні характеристики:

- однотактовий пристрій множення з накопиченням розрядністю 32 × 32;
- спеціалізований набір команд;
- апаратна підтримка виконання програмних циклів і швидкого повернення з переривань;
- розширена до 1 Кслів кеш-пам'ять команд;
- п'ять 32-розрядних адресних шин — внутрішні односпрямовані шини адреси *X* та *Y*, програмна адресна шина і дві зовнішні адресні шини;
- сім 32-розрядних шин даних — внутрішні двоспрямовані шини даних *X* і *Y*, внутрішня двоспрямована глобальна шина даних, внутрішня двоспрямована шина даних ПДП, внутрішня двоспрямована програмна шина даних і дві зовнішні шини даних;
- внутрішньокристальна пам'ять мікропроцесора М1024 слова програмної пам'яті (*RAM*), дві незалежні пам'яті даних по 512 слів кожна (*RAM*), дві незалежних ПЗУ ємності по 1024 слова і ПЗУ початкового завантаження ємністю 64 слова;
- зовнішня пам'ять процесора може становити по 2×2^{32} 32-розрядних слів для команд і даних;
- продуктивність мікропроцесора на тактовій частоті 40 МГц становить близько 200 *MIPS*.

Сигнальні процесори фірми Analog Devices утворюють два сімейства — ADSP21xx і ADSP210xx. Технічні характеристики процесорів наведено в табл. 8.1.

Сімейство мікропроцесорів ADSP21xx — набір однокристальних 16-розрядних мікропроцесорів із загальною базовою архітектурою, оптимізованою для виконання алгоритмів цифрового оброблення сигналів та інших додатків, що потребують

Таблиця 8.1. **Цифрові сигнальні процесори фірми Analog Devices**

Тип процесора	Тактова частота, МГц	Тривалість циклу, нс	Продуктивність, *MIPS*	Внутрішня пам'ять	Внутрішня пам'ять програм		Зовнішня пам'ять програм-даних	*IO* лінії	Послідовний порт	Таймери	ПДП
					RAM	ROM					
ADSP-2101	20	50	20	1K×16	2K×24	–	16K×16 16K×24	2	2	1	0
ADSP-2103	10,24	97,6	10,24	1K×16	2K×24	–	16K×16 16K×24	2	2	1	0
ADSP-2104	10,24	50	20	512×16	1K×24	–	16K×16 16K×24	2	1	1	0
ADSP-2104L	13,824	72,3	13,824	512×16	1K×24	–	16K×16 16K×24	2	1	1	0
ADSP-2105	10,24	100	20	256×16	512×24	–	16K×16 16K×24	2	1	1	0
ADSP-2109	10,24	50	20	512×16	1K×24	4K×24	16K×16 16K×24	2	1	1	0
ADSP-2109L	13,824	72,3	13,824	512×16	1K×24	4K×24	16K×16 16K×24	2	1	1	0
ADSP-2111	20	50	20	1K×16	2K×24	–	16K×16 16K×24	5	2	1	0

ADSP-2115	20	50	20	512×16	1K×24	–	16K×16 16K×24	2	2	1	0
ADSP-2161	16,67	60	16,67	1K×16	–	8K×24	16K×16 16K×24	2	2	1	0
ADSP-2162	10,24	100	10,24	1K×16	–	8K×24	16K×16 16K×24	2	2	1	0
ADSP-2163	16,67	60	16,67	512×16	–	4K×24	16K×16 16K×24	2	2	1	0
ADSP-2164	10,24	100	10,24	512×16	–	4K×24	16K×16 16K×24	2	2	1	0
ADSP-2165	20	50	20	4K×16	1K×24	12K×24	16K×16 16K×24	*N/A*	2	1	0
ADSP-2166	16,67	60	16,67	4K×16	1K×24	12K×24	16K×16 16K×24	*N/A*	2	1	0
ADSP-2171	16,67	30	33	2K×16	2K×24	–	16K×16 16K×24	5	2	1	0
ADSP-2172	16,67	30	33	2K×16	2K×24	8K×24	16K×16 16K×24	5	2	1	0
ADSP-2173	10,24	50	20	2K×16	2K×24	–	16K×16 16K×24	5	2	1	0

Продовж. табл. 8.1

Тип процесора	Тактова частота, МГц	Тривалість циклу, нс	Продуктивність, *MIPS*	Внутрішня пам'ять	Внутрішня пам'ять програм		Зовнішня пам'ять програм-даних	*IO* лінії	Послідовний порт	Таймери	ПДП
					RAM	ROM					
ADSP-2181	16,67	30	33	16K×16	16K×24	–	16K×16 16K×24	13	2	1	1
ADSP-2183	16,67	34,7	33	16K×16	16K×24	–	16K×16 16K×24	13	2	1	1
ADSP-21msp50	20	50	20	1K×16	2K×24	–	16K×16 16K×24	*N/A*	2	1	*N/A*
ADSP-21msp58	13	38	26	2K×16	2K×24	–	16K×16 16K×24	3	2	1	0
ADSP-21msp59	13	38	26	2K×16	2K×24	4K×24	16K×16 16K×24	3	2	1	0
ADSP-21csp01	25	20	50	4K×16	4K×24	–	16 Мслів	12	2	1	5
ADSP-21020	33,3	30	33,3	0	0	–	4 Гслів	*N/A*	2	1	*N/A*
ADSP-21060	40	25	40	4 Мбіт	–	–	4 Гслів	4	2+6	1	10
ADSP-21061	50	20	50	1 Мбіт	–	–	4 Гслів	4	2	1	6
ADSP-21061L	44	22,5	44	1 Мбіт	–	–	4 Гслів	4	2	1	6
ADSP-21062	40	25	40	2 Мбіт	–	–	4 Гслів	4	2+6	1	10
ADSP-21065L	60	16,67	60	544 Кбіт	–	–	64 Мслів	12	8	2	10

Рис. 8.3. Порівняння архітектур сигнальних процесорів

високошвидкісних обчислень з фіксованою крапкою. Сімейство нині налічує 14 представників, що відрізняються один від одного переважно розміщеними на кристалі периферійними пристроями, такими як кеш-пам'ять, таймери, порти тощо.

Друге сімейство мікропроцесорів ADSP210xx поєднує 32-розрядні мікропроцесори, орієнтовані на оброблення сигналів алгоритмів, що вимагають виконання обчислень з плаваючою комою. До сімейства належать мікропроцесори ADSP21010, ADSP21020, ADSP21060, ADSP21062.

Порівняння архітектур сигнальних процесорів за швидкодією. Основні показники продуктивності процесора ADSP-21061 з тактовою частотою 40 Мгц (тривалість циклу 25 нс) під час виконання тестових програм такі:

- швидке перетворення інтегралів Фур'є на 1024 відліки, 460 мкс (18 221 цикл);
- секція фільтра з кінцевою імпульсною характеристикою (*FIR*), 25 нс (1 цикл);
- секція фільтра з нескінченною імпульсною характеристикою (*IIR*), 100 нс (4 цикли)
- ділення (y/x), 150 нс (6 циклів)
- обчислення $1/\sqrt{x}$, 225 нс (9 циклів)
- швидкість передавання даних через канали *DMA*, 240 Мбайт/с.

На рис. 8.3. наведено порівняння архітектур сигнальних процесорів за кількості тактів, що потребують виконання типових ДСП алгоритмів — швидке перетворення інтегралів Фур'є (*Fourier*), фільтри *FIR* та *IIR*.

Як видно з діаграми, архітектура процесора DSP563xx фірми Motorola має більшу ефективність і забезпечує кращу тривалість виконання задач за рівних значеннях тактових частот мікропроцесора.

Контрольні запитання

1. Які сфери використання сигнальних процесорів?
2. Наведіть приклади задач цифрового оброблення сигналів.
3. Якими факторами визначається швидкодія сигнальних процесорів?
4. Які є класи сигнальних процесорів?
5. Назвіть основні блоки структурної схеми сигнального процесора оброблення даних з фіксованою комою.
6. У чому суть модифікації гарвардської архітектури в сигнальному процесорі TMS320XC5x?
7. Яке призначення має додатковий арифметичний пристрій *ARAU*?
8. Які функції виконує логічний блок *PLU*?
9. У чому відмінність пам'яті *SARAM* від пам'яті *DARAM*? Який тип пам'яті більш швидкодіючий?
10. Назвіть периферійні пристрої сигнального процесора.
11. Назвіть джерела переривань сигнального процесора.
12. В яких енергозберігаючих режимах може працювати сигнальний процесор?
13. Яке призначення генератора тактів очікування *S/W*?
14. Які функції виконує блок початкового завантаження *BOOT LOADER*?
15. Назвіть основні блоки структурної схеми сигнального процесора оброблення даних з плаваючою комою.
16. Які шини використовує сигнальний процесор TMS320C3x для незалежного доступу до вбудованої пам'яті програм і даних?
17. Які формати подання чисел є у сигнальному процесорі TMS320C3x? Яке найбільше та найменше число, з якими оперує МП?
18. Які переваги має обчислення у форматі з плаваючою комою порівняно з обчисленням чисел з фіксованою комою?
19. Яке призначення арифметичних пристроїв *ARAU*0 та *ARAU*1?
20. Які функції виконує контролер *DMA*?
21. Назвіть периферійні пристрої сигнального процесора TMS320C3x.
22. Назвіть джерела переривань сигнального процесора TMS320C3x.
23. В яких енергозберігаючих режимах може працювати сигнальний процесор TMS320C3x?
24. Які функції виконує блок початкового завантаження *BOOT LOADER*?
25. Дайте характеристику режиму адресації сигнального процесора.
26. Де використовуються сигнальні процесори з плаваючою комою?
27. В якому процесорі поєднано п'ять процесорів та як вони взаємодіють?
28. Назвіть основні галузі застосування процесорів Texas Instruments.
29. Який з процесорів має архітектуру VelociTI? Чим вона відрізняється від архітектури VLIW?
30. Які типи сигнальних процесорів виробляє фірма Motorola? Яке застосування кожної з цих груп?
31. Які типи сигнальних процесорів виробляє фірма Analog Devices?
32. Порівняєте за швидкодією сигнальні процесори фірм Texas Instruments, Motorola та Analog Devices для різних задач цифрового оброблення сигналів.

Розділ 9

НЕЙРОННІ ОБЧИСЛЮВАЧІ

9.1. Основні поняття і завдання нейронних обчислювачів

У попередніх розділах було розглянуто приклади розв'язання задач, які добре формалізовані, тобто для них розроблені математичні моделі і можуть бути застосовані алгоритми, що ґрунтуються на правилах типу «якщо А, то Б». Однак існує ряд задач, які важко формалізувати, тобто знайти чіткий алгоритм розв'язання. До таких задач належать:

розпізнавання зображень, наприклад, розпізнавання рукописних і друкарських символів під час оптичного введення в ЕОМ, розпізнавання типів клітин крові, розпізнавання мови. При цьому об'єкт, що розпізнається, є масивом даних, який треба віднести до одного із заздалегідь відомих класів;

кластерізація даних (пошук закономірностей). Вхідні дані слід віднести до будь-якої групи (кластеру) за властивою їм «близькістю», причому число кластерів заздалегідь невідоме. Як критерії «близькості» можуть бути використані відстань між векторами даних, значення коефіцієнта кореляції тощо;

апроксимація функцій. Знайти функцію, що апроксимує невідому, наприклад набір експериментальних даних. Ця задача актуальна під час моделювання складних систем і створення систем керування складними динамічними об'єктами, для робастного керування;

прогнозування. За попереднім поводженням функції спрогнозувати її поводження у майбутньому. Ця задача актуальна для керування системами з прогнозуванням та для систем прийняття рішень;

оптимізація. Мета цих задач — знайти оптимальне значення цільової функції, що задовольняє ряду обмежень.

Слід зазначити, що людина добре розв'язує задачі, які важко формалізувати, — розпізнає зображення, класифікує дані, прогнозує тощо. Тому ідея створення штучного розуму стала досить актуальною. Однак для цього треба було провести

численні дослідження принципів функціонування мозку людини з погляду оброблення інформації.

Мозок людини є найскладнішою з відомих систем переробки інформації. В ньому міститься близько 100 млрд нервових клітин, або нейронів, кожна з яких має в середньому 10 000 зв'язків.

Нейрони — особливий вид клітин, основне призначення яких полягає в оперативному керуванні організмом. Схематичне зображення нейрона наведено на рис. 9.1.

Нейрон має тіло (сому) *2*, дерево входів (дендрити) *1* і виходів (аксонів) *4*. Дендрити сильно розгалужуються, пронизуючи порівняно великий простір навколо нейрона. Початковий сегмент аксона — стовщений аксонний горбок *3*, що прилягає до тіла клітини. У міру віддалення від клітини він поступово звужується і на ньому з'являється мієлінова оболонка, що має високий електричний опір. На сомі та на дендритах розміщуються закінчення аксонів, що йдуть від інших нервових клітин. Кожне таке закінчення *5* має вигляд стовщення, яке називають синаптичною бляшкою, або синапсом. Вхідні сигнали дендритного дерева (постсинаптичні потенціали) зважуються і підсумовуються на шляху до аксонового горбка, де генерується вихідний імпульс. Його наявність (або інтенсивність) є функцією зваженої суми вхідних сигналів. Вихідний сигнал проходить по гілках аксона і досягає синапсів, що з'єднують аксони з дендритними деревами інших нейронів. Через синапси сигнал трансформується на новий вхідний сигнал для суміжних нейронів. Цей вхідний сигнал може бути позитивним і негативним (збудливим або гальмуючим) залежно від виду синапсів. Значення вхідного сигналу, що генерується синапсом, може відрізнятися від значення сигналу, який приходить у синапс. Ці розходження визначають ефективність, або вагу синапса. Синаптична вага може змінюватися у процесі функціонування синапса.

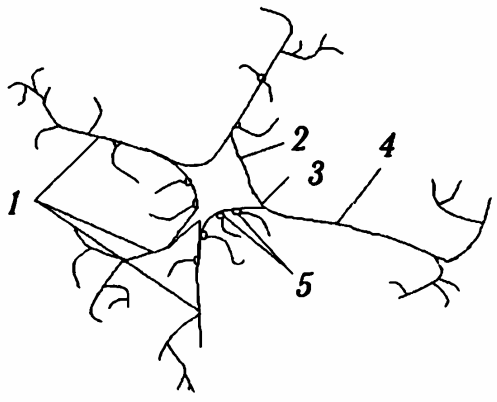


Рис. 9.1. Схематичне зображення нейрона

Учені різних спеціальностей робили спроби створити математичну модель нейрона. Так, біологи намагалися одержати аналітичне уявлення нейрона, що враховувало б усі його відомі функціональні характеристики. Однак основне завдання — передавання інформації нервовим імпульсом — втрачалося серед множини параметрів,

що належать до фізики провідності імпульсів. Тому спробували замінити фізичний опис нейрона логічним. При цьому нервова клітина розглядалася як елемент, що передає інформацію. У 1943 р. вчені-математики Мак-Каллох і Пітс зобразити нейрон як простий перемикальний елемент, що може знаходитися в одному із двох стійких станів «ввімкнене» або «вимкнене». Нейрон спрацьовує, якщо алгебраїчна сума входів у цей час більша, ніж порога. Нейрон у такому уявленні може використовуватися як елемент ЕОМ і дає змогу побудувати мережу з нейронів з відповідними порогами і зв'язками, що реалізовувала б довільну булеву функцію або таблицю істинності. Ці дослідження привели до численних винаходів схем оброблення інформації, пристроїв розпізнавання і сенсорних аналізаторів.

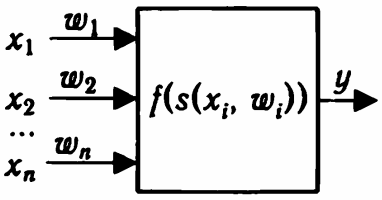


Рис. 9.2. Модель нейрона

Нині найчастіше використовують модель нейрона, зображену на рис. 9.2.

Нейрон має n односпрямованих входів (синапсів), з'єднаних з виходами інших нейронів та вихід y (аксон), по якому сигнал (збудження або гальмування) надходить на синапси наступних нейронів. Синапси характеризується значенням синаптичного зв'язку, або ваги w_i, що за фізичним змістом еквівалентний електричній провідності. Кожний нейрон характеризується своїм *поточним станом* s за аналогією з нервовими клітинами головного мозку, що можуть бути збуджені або загальмовані.

Поточний стан нейрона залежить від значення його входів, ваг та, можливо, попереднього стану. Найчастіше стан нейрона визначається або як зважена сума його входів

$$s = \sum_{i=1}^{n} x_i w_i, \tag{9.1}$$

або як відстань між вектором входів і вектором ваг входів

$$s = \sum_{i=1}^{n-x} |w_i - x_i|. \tag{9.2}$$

Вихід y нейрона є функцією його стану:

$$y = f(s). \tag{9.3}$$

Функцію $f(s)$ називають *функцією активації.*

Таблиця 9.1. **Функції активації нейрона**

Назва функції	Визначення
Східчаста порогова	$f(s) = \begin{cases} 0, & s < a \\ 1, & s \geq a \end{cases}$
Лінійна порогова	$f(s) = \begin{cases} 0, & s < a_1 \\ ks + b, & a_1 \leq s < a_2 \\ 1, & s \geq a_2 \end{cases}$
Сигмоїдна	$f(s) = (1 + e^{-k(s-a)})^{-1}$
Лінійна	$f(s) = ks + b$
Гауссіана	$f(s) = e^{-k(s-a)^2}$

Найпоширенішими функціями активації є східчаста порогова, лінійна порогова, сигмоїдна та лінійна і гауссіана, які наведені в табл. 9.1.

Нейронна мережа створюється внаслідок об'єднання виходів нейронів зі входами інших, причому нейрони створюють шари, які з'єднані між собою. *Нейронна мережа* — це мережа з кінцевим числом шарів, що складаються з однотипових елементів та різних типів зв'язків між шарами нейронів. При цьому кількість нейронів у шарах вибирають із розрахунку забезпечення заданої якості розв'язання задачі, а кількість шарів нейронів — якомога меншим для зменшення тривалості розв'язання.

Найпростішу одношарову нейронну мережу, яку ще називають також простим персептроном, наведено на рис. 9.3. На n входів надходять сигнали, які проходять по синапсах на три нейрони, що утворюють єдиний шар з вихідними сигналами $y_j = f\left[\sum_{i=1}^{n} x_i w_{ij}\right]$, де $j = 1...3$.

Двошаровий персептрон, отриманий з одношарового додаванням другого шару, що складається з двох нейронів, наведено на рис. 9.4. При цьому нелінійність активаційної функції має велике значення: якби її не було, результат функціонування будь-якої p-шарової нейронної мережі з ваговими матрицями $W^{(i)}$, $i = 1,2,...p$ для кожного шару i зводився б до перемножування вхідного вектора сигналів X на матрицю $W^{(S)} = W^{(1)} * W^{(2)} * ... * W^{(p)}$, тобто фактично така p-шарова нейронна мережа була б еквівалентна одношаровій з ваговою матрицею єдиного шару $W^{(S)}$: $Y = XW^{(S)}$.

Крім числа шарів і зв'язків між ними, нейронні мережі класифікуються як ациклічні або циклічні. Наведені на рис. 9.3 та 9.4 приклади належать до ациклічних нейронних мереж. На рис. 9.5 наведено приклад циклічної нейронної мережі.

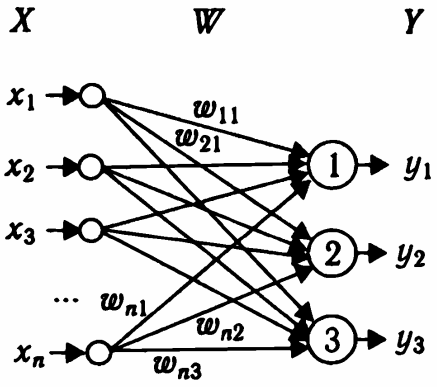


Рис. 9.3. Одношаровий персептрон

Якщо розглянуті схеми (див. рис. 9.3—9.5) доповнити положенням про тактування мережі (задати тривалість спрацювання нейронів), то отримаємо апарат для завдання різних алгоритмів оброблення даних за допомогою нейронних мереж, які можна використати для розв'язання як формалізованих задач, так і задач, які важко формалізувати. В останньому випадку застосування нейронних мереж ґрунтується не на виконанні запропонованого алгоритму, а на запам'ятовуванні мережею поданих їй прикладів на етапі створення мережі та виробітку результатів, погоджених з цими прикладами, на етапі розв'язання задачі.

За типом сигналів у нейронній мережі останні поділяють на *бінарні* (цифрові) й *аналогові*. Бінарні оперують двійковими сигналами, і вихід кожного нейрона може набувати лише два значення — 0 або 1.

За можливістю адаптації можна виділити: нейронні мережі, що *конструюють* та *навчають*. У мережі, що конструю-

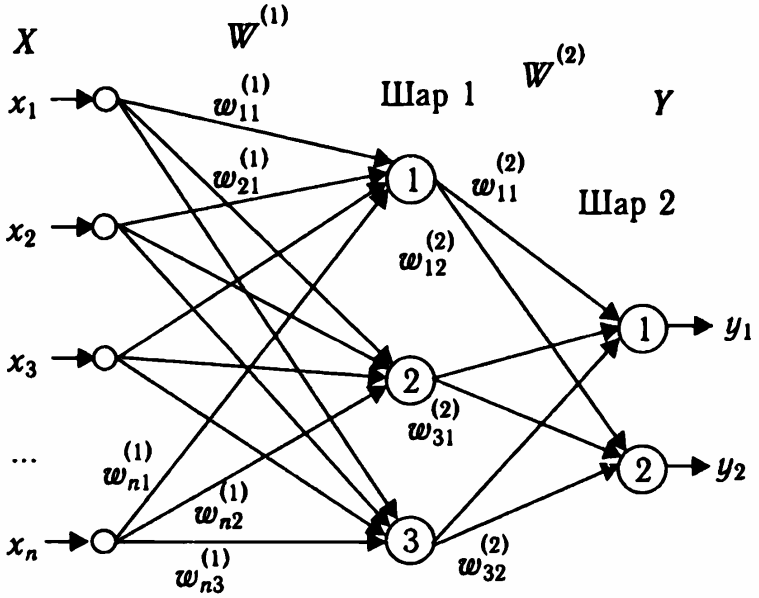


Рис. 9.4. Двошаровий персептрон

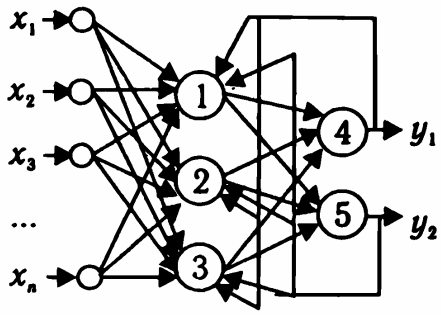


Рис. 9.5. Циклічна нейронна мережа

ють, задають число і тип нейронів, графи міжнейронних зв'язків, вагу входів, а в мережі, що навчають, — графи міжнейронних зв'язків та ваги входів, що змінюються під час виконання алгоритму навчання.

За алгоритмом навчання мережі поділяють на мережі, за якими *спостерігають, не спостерігають* і *змішані* (гібридні). Перші у процесі навчання порівнюють заздалегідь відомий результат з отриманим. Другі навчаються, не знаючи правильних значень результату. Вони групують вхідні дані так, щоб вони формували той самий вихід мережі. Такий підхід використовують, наприклад під час розв'язання задачі кластеризації. За змішаного алгоритму навчання частина ваг визначається під час спостереження, а частина — без спостереження.

9.2. Основи побудови алгоритмів навчання нейронних мереж

Одним з найважливіших етапів розробки нейронного обчислювача є навчання мережі. Від якості навчання залежить спроможність мережі вирішувати поставлені перед нею задачі. На етапі навчання крім параметра якості добору вагових коефіцієнтів велике значення має тривалість навчання. Як правило, ці два параметри зв'язані зворотною залежністю і їх доводиться вибирати на основі компромісу.

Навчання починається з вибору вихідної мережі (евристично обраний граф) із заданою кількістю входів і виходів. Наприклад, у літературі рекомендується тришарова мережа з числом нейронів внутрішнього шару, що дорівнює півсумі числа входів і виходів мережі. Кожний нейрон внутрішнього шару зв'язаний з виходами всіх вхідних нейронів мережі, причому кожний вихідний нейрон зв'язаний з виходами всіх нейронів внутрішнього шару. Потім вибирають ваги входів нейронів мережі так, щоб відбувалося розв'язання задачі. Якщо це не вдається, то змінюють граф мережі.

Найпоширенішим алгоритмом визначення ваг для навчання зі спостереженням персептронних мереж, для якого доказано збіг процесу, є *алгоритм зворотного поширення помилки.* Нині його реалізують 80 % нейрочипів, орієнтованих на розв'язання задач цифрового оброблення сигналів. Крім того, він став

деяким еталоном для встановлення продуктивності нейронних обчислювачів так само, як швидке перетворення інтегралів Фур'є на 1024 відліків для сигнальних процесорів (див. п. 8.3).

Під час навчання сигнал помилки поширюється по мережі у зворотному напрямі. Виконується корекція ваг входів нейронів, що запобігає повторній появі цієї помилки.

Алгоритм навчання *одношарового персептрона* (див. рис. 9.2 або рис 9.3) такий. Існує набір прикладів (X_1, D_1), (X_2, D_2), ..., (X_m, D_m), де $X_j = \{x_{j1}, x_{j2} ... x_{jn}\}$ — вхідні значення j прикладу, D_j — вихідне значення цього прикладу. Вважають, що правильно навчений персептрон, якщо для всіх j $\max|D_j - Y_j| \le \delta$, де δ — задане значення помилки.

1. Присвоїти вагам і порогу нейрона випадкові малі значення.

2. Подати на входи нейрона черговий приклад $\langle X_j \rangle$, починаючи з першого приклада, і визначити значення виходу нейрона Y_j, де $j = 1, ..., m$.

3. Змінити ваги згідно з виразом $w_i(t+1) = w_i(t) + a(D_j - Y_j)x_i$, $i = 1, 2, ...n$, a — коефіцієнт, $0 < a < 1$.

4. Проводити кроки 2,3 доти, поки помилка на всіх прикладах не буде перевищувати заданого значення δ.

У *багатошарових мережах* алгоритм залишається тим самим, за винятком кроку 2, який складається з етапів корекції шарів, причому корекція починається з вихідного шару.

Крім багатошарових нейронних мереж зворотного поширення, основним недоліком яких є неможливість гарантувати найкраще навчання за певний часовий інтервал, існує досить велика кількість інших варіантів побудови нейронних мереж із певними перевагами та недоліками.

Проектування нейросистем — складний і трудомісткий процес, в якому вибір відповідного алгоритму — лише один з кількох кроків процесу проектування. Він, як правило, включає: дослідження предметної галузі, структурно-функціональне проектування, топологічне проектування, тощо.

9.3. Апаратна реалізація нейронних обчислювачів

Розрізняють такі архітектури обчислювальних систем:

- з одним потоком команд і даних *SISD (Single Instruction — Single Data)*, рис. 9.6, *а*;
- із загальним потоком команд і множинним потоком даних *SIMD (Single Instruction —Multy Data)*, рис. 9.6, *б*;

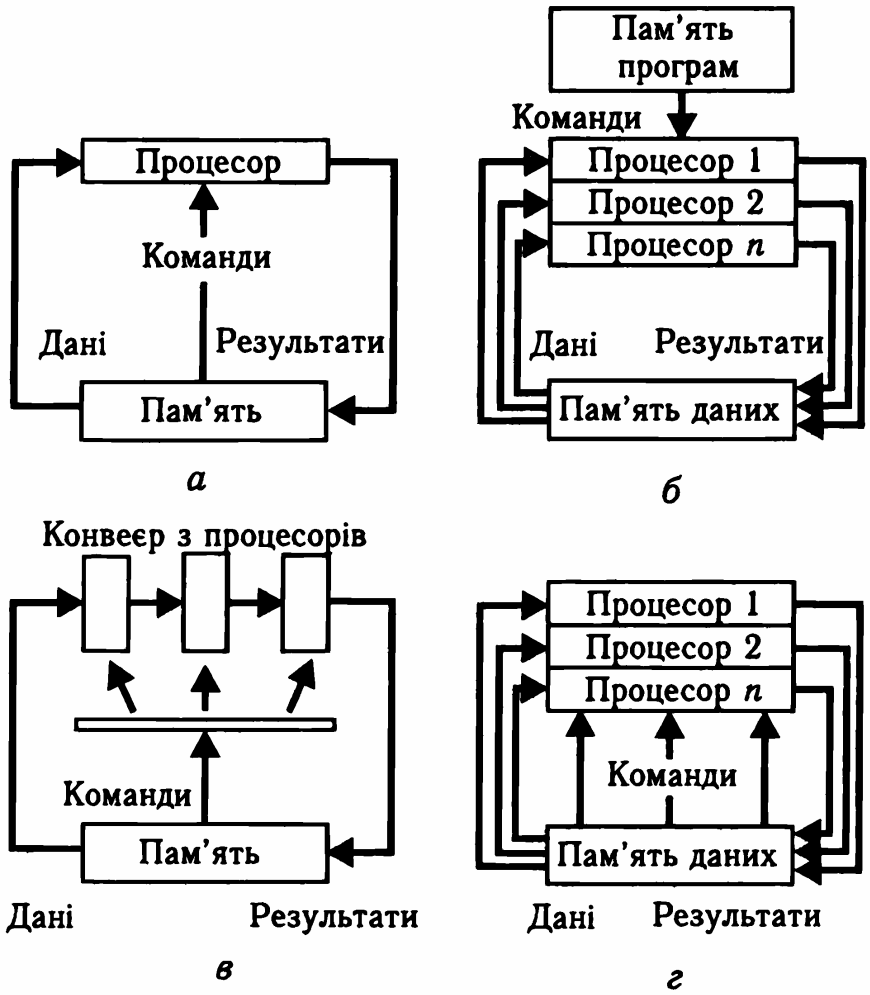


Рис. 9.6. Архітектури обчислювальних систем:
а — SISD; б — SIMD; в — MISD; г - MIMD

• із множинним потоком команд та одним потоком даних *MISD (Multy Instruction — Single Data)*, рис. 9.6, *в*;

• із множинними потоками команд і даних *MIMD (Multy Instruction — Multy Data)*, рис. 9.6, *г*;

Нейронний обчислювач — система з *MIMD* архітектурою, яка працює за алгоритмом нейронної мережі. Нейрообчислювачі належать до масово-паралельних систем, що характеризуються паралельними потоками однакових команд і множинного потоку даних.

Існує три основних напрями побудови нейронних обчислювачів:

• на базі каскадного з'єднання універсальних *RISC* або *CISC* мікропроцесорів фірм Intel, AMD, Sparc, Alpha, Power PC, MIPS;

• на базі програмних логічних матриць *PLM* або процесорів з паралельним обробленням даних на апаратному рівні, наприклад, сигнальних процесорів фірм TMS, ADSP, Motorola;

• на спеціалізованій елементній базі — однобітових процесорів і нейрочипів.

Системи першого напряму називають *нейроемуляторами.* Їхня апаратна реалізація ґрунтується на використанні універсальних *RISC* або *CISC* мікропроцесорів. Нейроемулятори реалізують типові *нейрооперації* (обчислюють зважену суму, виконують нелінійні перетворення) на програмному рівні.

Системи другого і третього напрямів у вигляді плат розширення стандартних обчислювальних систем називають *нейроприскорювачами,* а системи третього напряму у вигляді функціонально закінчених обчислювальних пристроїв — *нейрокомп'ютерами.*

Нейроприскорювачі поділяють на два класи — *«віртуальні»,* що вставляються у слот розширення стандартного комп'ютера, і *«зовнішні»,* що з'єднуються з керуючою *Host* ЕОМ.

У нейроприскорювачах на базі програмних логічних матриць PLM алгоритми нейромережі реалізовано апаратно. Сучасні *PLM* мають значний об'єм ресурсів (наприклад, *PLM* фірми Xilinx має на кристалі до 4 млн системних вентилів) і можуть апаратно реалізовувати структуру нейромережі, зображену на рис. 9.4, 9.5. Характеристики *PLM* для реалізації нейрообчислювачів подано в табл. 9.2.

Таблиця 9.2. **Особливості реалізації нейрообчислювачів на *PLM***

Тип *PLM*	Виробник	Кількість нейронів
XC4005E/XL	XILINX	6
XC4013XLA	XILINX	18
XC4020XLA	XILINX	24
XC4044XLA	XILINX	50
XC4062XLA	XILINX	72
XC4085XL	XILINX	97
XC40250XV	XILINX	200
EPF10K20	ALTERA	4
EPF10K50E	ALTERA	11
EPF10K100E	ALTERA	19
EPF10K250E	ALTERA	50
M4LV-96/48	AMD	3
M4LV-192/96	AMD	6
M5LV-256	AMD	8
M5LV-512	AMD	16

Обчислювальні системи на базі *PLM* характеризуються досить високими частотами роботи, але зміна алгоритмів роботи потребує перепрограмування ВІС. Нейроприскорювачі на базі *PLM* нині використовують як гнучкі нейрообчислювальні системи з науково-дослідницькою метою та для дрібносерійного виробництва.

Для побудови продуктивніших нейрообчислювачів, як правило, застосовують *сигнальні процесори* (див. розд. 8). Сигнальні мікропроцесори, які розроблялися для цифрового оброблення сигналів, як виявилося, спроможні ефективно інтерпретувати алгоритми нейромереж. Вони орієнтовані на оброблення масивів (векторів) даних і виконують операцію множення з накопиченням. Деякі з них можуть одночасно виконувати дві команди. Ці особливості дають змогу легко реалізувати множення з накопиченням векторів ваг і векторів входів нейронів мережі. Звичайно під час створення нейрообчислювачів використовують гібридну структуру, коли блок матричних обчислень реалізується на базі каскадного з'єднання сигнальних процесорів, а логіка керування — на основі *PLM.*

Схему *зовнішнього нейроприскорювача,* що з'єднується з керуючою *Host*-ЕОМ і виконана на базі каскадного з'єднання сигнальних процесорів, зображено на рис. 9.7.

Керуюча *Host-EOM* реалізована на основі звичайної обчислювальної системи з *CISC*- або *RISC*-мікропроцесорами.

Модуль матричних сигнальних процесорів об'єднує їх між собою відповідно до структури нейромережі. Схема містить також робочу пам'ять, пам'ять програм, модуль введення-виведення сигналів (включає АЦП, ЦАП і *TTL* лінії), а також

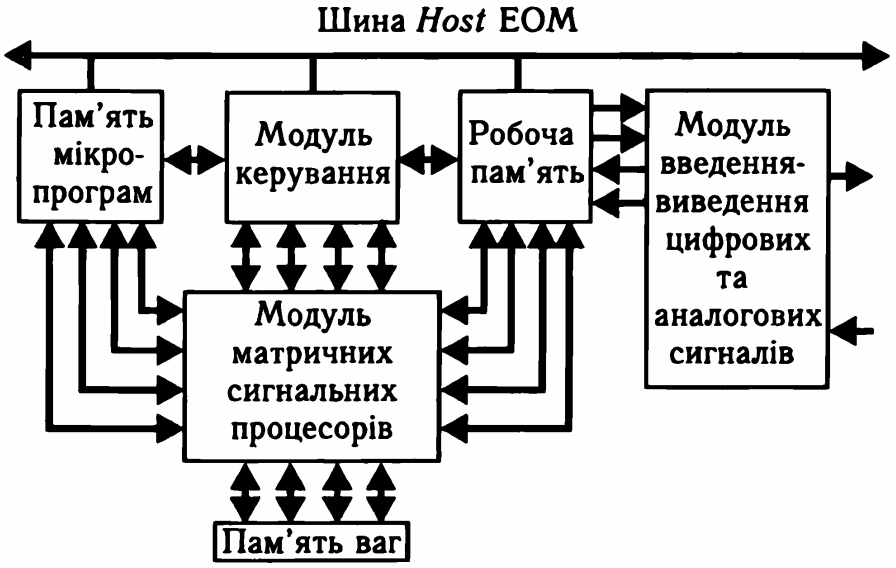


Рис. 9.7. Узагальнена функціональна схема віртуального нейроприскорювача

модуль керування, що може бути реалізований на основі спеціалізованого керуючого сигнального процесора, на основі *PLM* або мати роздільну структуру, за якої функції загального керування розподілені між матричними сигнальними процесорами. Для побудови нейроприскорювача цього типу найперспективнішим є використання сигнальних процесорів з плаваючою комою — ADSP2106x, TMS320C4x,8x, DSP96002 та ін. Вибір того або іншого сигнального процесора — багатокритеріальна задача, однак слід зазначити перевагу процесорів Analog Devices для додатків, що потребують виконання великих обсягів математичних обчислень, таких як цифрова фільтрація сигналу, обчислення кореляційних функцій. Для розв'язання задач, що потребують інтенсивного обміну із зовнішніми пристроями, доцільно використовувати процесори Texas Instruments, обладнані високошвидкісними інтерфейсними підсистемами. Компанія Motorola є лідером щодо обсягу виробництва дешевих і досить продуктивних 16- і 24-розрядних сигнальних мікропроцесорів з фіксованою точкою.

Елементною базою нейрообчислювачів третього напряму, тобто нейрокомп'ютерів, є *нейрочипи*. Нейрочипи бувають цифрові, аналогові та гібридні. Вони також можуть мати схеми настроювання ваг під час навчання або можуть не мати таких схем і передбачати зовнішнє завантаження ваг. Характеристики деяких поширених нейрочипів наведено у табл. 9.3.

У табл. 9.3 дано оцінку продуктивності нейрообчислювачів замість традиційних *(M)IPS* і *(M)FLIPS*-(мільйонів) операцій з фіксованою і плаваючою точкою, проводиться інший показник: *CPS (connections per second)* — число з'єднань за секунду. *З'єднання* — це множення значень входу на вагу і додавання з накопиченням.

Оскільки цей показник не враховує розрядності входів і ваг, іноді приводять такі показники:

- $CPSPW = CPS/N_w$, де N_w — число синапсів у нейроні;
- *CPPS* — число з'єднань примітивів за секунду, $CPPS = CPS \times B_w \times B_s$, де B_w і B_s відповідно розрядність ваг і синапсів.

Показником, що оцінює швидкість навчання, є *CUPS* (*connections update per second*) — число змінених значень ваг за секунду.

Цифрові нейрочипи. Одним із перших нейрочипів була BIC MD1220 фірми Micro Device. Цей кристал інтепретує 8 нейронів і 8 зв'язків із 16- розрядними вагами, що зберігаються у внутрішньокристальній пам'яті, та одноразрядними входами, які мають послідовні помножувачі. Тривалість такту становить 7,2 мкс, що забезпечує продуктивність 9 *MCPS*.

Таблиця 9.3. **Характеристики нейрочипів**

Тип, фірма	Максимальна кількість			Розрядність		Продуктивність			
	синапсів	нейронів	шарів	входів	ваг	CPS	CPSPW	CPPS	CUPS
	Цифрові								
MD1220 Micro Device	64	8	8	1	16	9M	1M	142M	–
NLX420 Adaptive Logic	1M	16	16	16	16	10M	20K	640M	–
Lneuro 1,0 Philips	1536	16	192	16	16	26M	26K	1,6Г	32M
	Аналогові								
80170NW Intel	2 банка 64 × 80	64	2	–	–	2 Г	30M	–	–
	Гібридні								
CLNN-32 Bellcore	406	32	1	–	5	2 Г	5M	–	–
CLNN-64 Bellcore	1024	64	1	–	5	2 Г	2M	–	–
ANNA AT&T	4096	256	1	–	6	2,1 Г	5,1K	–	–

Із цих нейрочипів унаслідок їх каскадного з'єднання можна побудувати нейрокомп'ютери.

Фірма Adaptive Logic випускає нейрочип NLX420 із шістнадцятьма процесорними елементами (ПЕ), кожний з яких має 32-розрядний суматор. Ваги і входи завантажуються як 16-розрядні слова, але можуть бути використані як шістнадцять одноразрядних слів, або як чотири чотирирозрядних, або як два восьмирозрядних, або як одне 16-розрядне слово. Ваги зберігаються поза чипом. Вхід загальний для всіх ПЕ, що дає змогу виконувати паралельно до 16 операцій множення. Функції активації задаються користувачем. Кристали можна з'єднувати каскадно.

Кристал Lneuro 1,0 фірми Philips містить 16 ПЕ,кожний з яких може функціонувати як шістнадцять одноразрядних, вісім дворозрядних, чотири чотирирозрядних, два восьмирозрядних або один шістнадцятирозрядний. Чип має 1 Кбайт пам'яті ваг, що дає змогу використовувати 1024 8-розрядних або 512 16-розрядних вагових коефіцієнтів. Функція активації реалізується поза чипом, що сприяє під час каскадування інтерпретувати великі мережі.

Аналогові нейрочипи використовують аналогові схеми — суматори з аналоговими входами, ваги яких теж задаються аналоговим способом. Ці чипи зазвичай менші та простіші, ніж цифрові. Крім того, забезпечення необхідної точності вимагає ретельного проектування і виготовлення. Кристал фірми Intel 80170NW містить 64 нейрони і 2 банки 64 × 80 ваг. Можливо кілька мережних конфігурацій. Чип має 64 аналогових входи (0...3 В) і 16 внутрішніх зсувів. На кристалі можна реалізувати двошарову мережу з 64 входами, 64 внутрішніми і 64 вихідними нейронами. Інші конфігурації мають тришарові або одношарову мережі з 128 входами. Точність ВІС становить 5—6 розрядів для ваг і виходів.

Гібридні нейрочипи використовують комбінацію аналогового і цифрового підходів. Наприклад, входи можуть бути аналоговими, ваги завантажуватися як цифрові, а виходи бути цифровими.

Чипи CLNN-32, CLNN-64 фірми Bellcore містять 32 нейрони. Входи, виходи і внутрішнє оброблення сигналів — аналогові, а п'ятирозрядні ваги — цифрові.

Чип ANNA фірми AT&T переважно цифровий, але всередині має конденсаторні заряди для збереження ваг. Чип містить 4096 ваг. Число нейронів варіюється від 16 до 256 із числом входів у нейрон відповідно 256 або 16. Ваги мають точність шість розрядів.

Існують нейрочипи, в яких використовується подання даних частотою або шириною імпульсів.

Контрольні запитання

1. Для розв'язання якого класу задач використовують нейропроцесори? Назвіть приклади таких задач.
2. У чому полягає загальна суть використання нейронних мережевих обчислень?
3. Дайте визначення понять «нейрон» і «нейронна мережа».
4. Дайте визначення та наведіть приклади функцій стану нейрона.
5. Дайте визначення та наведіть приклади функцій активізації нейрона.
6. Наведіть приклади одно- та двошарових нейронних мереж.
7. Чим відрізняються циклічні нейронні мережі від ациклічних?
8. Які є типи нейронних мереж?
9. У чому полягає процес навчання нейронної мережі?
10. Назвіть основні кроки алгоритму зворотного поширення помилки для одношарових персептронів.
11. У чому полягає відмінність алгоритму зворотного поширення помилки для одно- і багатошарових персептронів?
12. Назвіть типи і наведіть приклади обчислювальних систем.
13. До якого типу обчислювальних систем належать нейронні обчислювачі?
14. За якими показниками оцінюють продуктивність нейромереж?
15. Як порівняти продуктивність традиційних фоннеймановських комп'ютерів і нейромереж?
16. Які існують основні напрями побудови нейронних обчислювачів?
17. Дайте визначення нейроемулятора, нейроприскорювача, нейрокомп'ютера. Порівняйте ці пристрої за апаратною реалізацією та швидкодією.
18. Перелічіть основні типи нейрочипів.
19. Які особливості сигнальних процесорів, що підвищують ефективність інтерпретації нейромережевих алгоритмів?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. *Абель П.* Язык Ассемблера для IBM PC и программирование / Пер. с англ. Ю. В. Сальникова. — М.: Высш. шк., 1992. — 447 с.
2. *Алексенко А. Г., Галицын А. А., Иванников А. Д.* Проектирование радиоэлектронной аппаратуры на микропроцессорах. — М.: Радио и связь, 1984. — 272 с.
3. *Балашов Е. П., Григорьев В. Л., Петров Г. А.* Микро- и мини-ЭВМ. — Л.: Энергоатомиздат, 1984. — 376.
4. *Вершинин О. Е.* Применение микропроцессоров для автоматизации технологических процессов. — Л.: Энергоатомиздат, 1986. — 208 с.
5. *Власов А. И.* Аппаратная реализация нейровычислительных управляющих систем // Приборы и системы управления. — 1999. — № 2. — С. 61—65.
6. *Галушкин А. И.* Некоторые исторические аспекты развития элементной базы вычислительных систем с массовым параллелизмом (80—90-е годы) // Нейрокомпьютер. — 2000. — № 1. — С. 68—82.
7. *Гольденберг А. М., Малев В. А., Малько Г. Б.* Цифровые устройства и микропроцессорные системы: Задачи и упражнения. — М.: Радио и связь, 1993. — 256 с.
8. *Горбунов В. Д., Панфилов Д. И., Преснухин Д. Л.* Микропроцессоры. Основы построения микро-ЭВМ. — М.: Высш. шк., 1984. — 144 с.
9. *Гребнев В. В.* Однокристальные микро-ЭВМ семейства AT89 фирмы Atmel. — СПб.: ЭФО, 1998. — 76 с.
10. *Гук М.* Процессоры Intel: от 8086 до Pentium II. — СПб: Питер, 1997. — 224 с.
11. *Гук М.* Процессоры Pentium II, Pentium Pro и просто Pentium. — СПб: Питер Ком, 1999. — 228 с.
12. *Евстигнеев А. В.* Микроконтроллеры AVR семейства Classic фирмы «ATMEL». — М.: ДОДЭКА-XXI, 2002. — 228 с.
13. *Козаченко В. Ф.* Микроконтроллеры: руководство по применению 18-разрядный микроконтроллеров Intel MCD-196/296 во встроенных системах управления. — М.: ЭКОМ, 1997. — 688 с.
14. *Корнеев В. В., Киселев А. В.* Современные микропроцессоры. — М.: НОЛИДЖ, 2000. — 320 с.
15. *Лебедев О. Н.* Микросхемы памяти и их применение. — М.: Радио и связь, 1990. — 234 с.
16. *Левенталь Л.* Введение в микропроцессоры. Программное обеспечение, аппаратные средства, программирование. — М.: Энергоатомиздат, 1983. — 464 с.
17. *Липовецкий Г. П., Литвинский Г. В., Оксинь О. Н.* Однокристальные микроЭВМ. Семейство МК48, Семейство МК51. Техническое описание и руководство по применению. — М.: МП «Бином», 1992. — 339 с.
18. *Лю Ю-Чжен, Гибсон Г.* Микропроцессоры семейства 8086/8088. Архитектура, программирование и проектирование микрокомпьютерных систем: Пер. с англ. — М.: Радио и связь, 1987. — 512 с.
19. *Майоров В. Г., Гаврилов А. И.* Практический курс программирования микропроцессорных систем. — М.: Машиностроение, 1989. — 215 с.
20. *Микроконтроллеры*: Сб. — М.: ДОДЭКА, 1998. — Вып. 1. — 384 с.
21. *Микропроцессорный* комплект К1810: Справочная книга. — М.: Высш. шк., 1990. — 269 с.
22. *Морс С. П., Альберт Д. Д.* Архитектура микропроцессора 80286. — М.: Радио и связь. — 1990. — 304 с.
23. *Самофалов К. Г., Виктров О. В., Кузняк А. К.* Микропроцессоры. — К.: Техніка, 1986. — 278 с.
24. *Справочник* по микропроцессорным устройствам / А. А. Молчанов, В. И. Корнейчук, В. П. Тарасенко и др. — К.: Техніка, 1987. — 228 с.
25. *Сташин В. В., Урусов А. В., Мологонцева О. Ф.* Проектирование цифровых устройств на однокристальных микроконтроллерах. — М.: Энергоатомиздат, 1990. — 221 с.

Навчальне видання

Схемотехніка ЕЛЕКТРОННИХ СИСТЕМ

У трьох книгах
Книга 3

Бойко Віталій Іванович
Гуржій Андрій Миколайович
Жуйков Валерій Якович
Зорі Анатолій Анатолійович
Петергеря Юлія Сергіївна
Співак Віктор Михайлович
Терещенко Тетяна Олександрівна
Якименко Юрій Іванович

МІКРОПРОЦЕСОРИ ТА МІКРОКОНТРОЛЕРИ

2-ге видання, доповнене і перероблене

Оправа і титул *В. С. Жиборовського*
Художній редактор *Г. С. Муратова*
Технічний редактор *А. І. Омоховська*
Коректори: *Т. М. Глушко, Р. Б. Попович*
Комп'ютерна верстка *Н. П. Довлетукаєвої*

Підп. до друку 26.04.2004. Формат 84 × 108/32. Папір. офс. № 1.
Гарнітура Peterburg. Офс. друк. Ум. друк. арк. 21,00.
Обл.-вид. арк. 24,24. Тираж 3000 пр. Вид. № 10536. Зам. № 4-163

Видавництво «Вища школа», 01054, Київ-54, вул. Гоголівська, 7 г

Свідоцтво про внесення до Держ. реєстру
від 04.12.2000 серія ДК № 268

Надруковано з плівок, виготовлених у видавництві «Вища школа»,
у ВАТ «Білоцерківська книжкова фабрика»,
09117, Біла Церква, вул. Л. Курбаса, 4