Міністерство транспорту та зв'язку України
Державний департамент з питань зв'язку та інформатизації України
ОДЕСЬКА НАЦІОНАЛЬНА АКАДЕМІЯ ЗВ'ЯЗКУ ім. О.С. ПОПОВА

Кафедра обчислювальної техніки та мікропроцесорів

Антонов О.С., Хіхловська І.В.

ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА МІКРОПРОЦЕСОРИ

Навчальний посібник
з дисципліни
"Обчислювальна техніка та мікропроцесори"
напряму Телекомунікації

Частина 1

ЗАТВЕРДЖЕНО
Методичною радою факультету
телекомунікаційних систем.
Протокол № 4
від 27 листопада 2007 р.

Одеса 2008

УДК 004 + 00431 + 621.39 (075) План НМВ 2007/2008 навч. р.

Антонов О.С., Хіхловська І.В. Обчислювальна техніка та мікропроцесори. Навчальний посібник. Частина 1. – Одеса: Вид. центр ОНАЗ ім. О.С. Попова, 2008. – 262 с.: іл.

Навчальний посібник призначено для самостійної роботи студентів з дисципліни “Обчислювальна техніка та мікропроцесори”, яка викладається за модульним принципом та має чотири модулі:

1 Вузли обчислювальної техніки та мікропроцесорних систем.

2 Програмування мікропроцесорів фірми Intel.

3 Мікропроцесорні системи на мікропроцесорах фірми Motorola та їх програмування.

4 Мікропроцесорні системи на мікроконтролерах, DSP фірми Motorola та їх програмування.

Розглянуто основні принципи побудови та функціонування обчислювальних та мікропроцесорних систем, їх основні вузли, у тому числі мікропроцесори. На прикладах мікропроцесорів, мікроконтролерів фірм Intel та Motorola показані принципи проектування мікропроцесорних систем, у тому числі, для цифрового оброблення сигналів та їх програмування. У посібнику наведено приклади застосування мікропроцесорів та мікроконтролерів різних моделей у пристроях телекомунікацій. Кожний розділ супроводжується запитаннями вхідного та вихідного контролю.

СХВАЛЕНО
на засіданні кафедри
обчислювальної техніки
та мікропроцесорів
і рекомендовано до друку.
Протокол № ____
від ___._____.200__р.

ЗМІСТ

ВСТУП

На етапі розвитку сучасних інформаційних мереж нового покоління уже неможливо собі уявити телекомунікаційне обладнання без сучасних мікропроцесорів та мікроконтролерів. Широкий спектр функцій, які реалізують системи комутації, шлюзи, маршрутизатори, інтегровані платформи, сервери, робочі станції, вимагає від процесорів та мікроконтролерів високої продуктивності та багатофункційності. Десятки років можна було спостерігати процес взаємного стимулювання розвитку процесорів, з одного боку, та побудованого на їх основі телекомунікаційного обладнання, з іншого.

Навчальний посібник призначено для самостійної роботи студентів напряму Телекомунікації з дисципліни “Обчислювальна техніка та мікропроцесори”. Дисципліна має 216 годин і складається з чотирьох модулів:

1 Вузли обчислювальної техніки та мікропроцесорних систем.

2 Програмування мікропроцесорів фірми Intel.

3 Мікропроцесорні системи на універсальних мікропроцесорах та їх програмування.

4 Мікропроцесорні системи на мікроконтролерах і DSP та їх програмування.

За вивченням першого модуля студенти отримують такі знання та уміння: подавати та трактувати вхідні та вихідні чисельні дані для подальшого цифрового оброблення. Співвідносити логічні змінні та функції з цифровими сигналами, що їх реалізують. Синтезувати цифрові пристрої, використовуючи типові цифрові блоки, вузли та елементи. Ставити та розв’язувати задачі, пов’язані з вибором засобів обчислювальної техніки, мікропроцесорів та мікроконтролерів за їх технічними, експлуатаційними та економічними характеристиками для систем телекомунікацій.

За вивченням другого модуля студенти отримують такі знання та уміння: створювати та налагоджувати програмне забезпечення для мікропроцесорів.

За вивченням третього модуля студенти отримують такі знання та уміння: ставити та розв’язувати задачі, пов’язані з аналізом, розробленням та експлуатацією мікропроцесорних систем у складі інформаційних та телекомунікаційних систем і мереж, створенням та налагодженням програмного забезпечення до них. Аналізувати та розробляти окремі вузли систем телекомунікацій, які використовують засоби обчислювальної техніки, мікропроцесори та мікроконтролери.

За вивченням четвертого модуля студенти отримують такі знання та уміння: створювати та налагоджувати програмне забезпечення для пристроїв управління, комутації, оброблення цифрових сигналів у системах телекомунікацій мовами конкретних мікропроцесорів та мікроконтролерів.

Посібник відрізняється від уже раніше виданих тим, що вміщує розділи, присвячені застосуванню мікропроцесорів в обладнанні телекомунікацій, створенню програмного забезпечення для пристроїв та систем телекомунікацій.

Знання архітектури сучасних мікропроцесорів та їх основних характеристик дасть можливість майбутнім фахівцям вибирати апаратуру

інформаційних мереж та систем з урахуванням можливостей застосовуваних у ній засобів обчислювальної техніки та мікропроцесорів, а також проектувати цю апаратуру на сучасному рівні.

Для розуміння викладаного у посібнику матеріалу студенти повинні знати такі теми з попередньо вивчених дисциплін:

І З дисципліни **“Основи схемотехніки”**:

1 Усі розділи цифрової техніки: логічні елементи, таблиці істинності, що описують їх роботу, шифратори, дешифратори, мультиплексори, демультиплексори, перемикальні функції, що описують їх роботу, регістри, лічильники імпульсів та таблиці переходів, що їх описують.

2 Інтегральна схемотехніка, технологія МОП та КМОП, великі (ВІС) та надвеликі (НВІС) інтегральні схеми, конструктивна реалізація.

3 Класифікація систем пам’яті: постійні запам’ятовувальні пристрої, ОЗП (статичний та динамічний), принципи зберігання інформації, ПЛІС, ВІС пам’яті, адресні, інформаційні та керувальні сигнали, що подаються на ВІС пам’яті, доступ до пам’яті, ємність ВІС пам’яті.

ІІ З дисципліни **“Інформатика”**:

1 Призначення обчислювальних систем, задачі, які можуть розв’язувати обчислювальні системи.

2 Подання даних, які обробляються в обчислювальних системах: подання даних у різних системах числення, подання даних з плаваючою та фіксованою точками, подання даних зі знаком.

3 Алгоритмізація задач, які розв’язуються обчислювальними системами, складання структурних схем алгоритмів розв’язуваних задач.

4 Мови високого рівня Delphi, C^{++}.

5 Мати навички складання та налагодження програм, які мають розгалуження та цикли, написаних мовами високого рівня.

ІІІ З дисципліни **“Дискретна математика”**:

1 Позиційні системи числення – двійкова, десяткова, шістнадцяткова, двійкова-десяткова, – перехід від одної системи числення до іншої.

2 Алгебра логіки – поняття логічної змінної та логічної функції, закони алгебри логіки, мінімізація логічних функцій методом Квайна та методами координатних діаграм.

1 МОДУЛЬ
1 ОБЧИСЛЮВАЛЬНІ ТА МІКРОПРОЦЕСОРНІ СИСТЕМИ

1.1 Основні визначення

Вхідний контроль:

1 Які засоби обчислювальної техніки Ви знаєте?
2 Які задачі Ви розв'язували за допомогою цих засобів?
3 Якими мовами програмування Ви пишете програми?
4 Опишіть відомі Вам великі інтегральні схеми (ВІС).
5 За якими технологіями виготовлені відомі Вам ВІС?

Обчислювальна техніка (ОТ) (computer science, computing machinery) – це систематизована сукупність наукових дисциплін і галузей техніки, що досліджує обчислювальні машини, принципи їх побудування і використання; займається розробленням і тестуванням апаратних засобів для оброблення і зберігання інформації; архітектури обчислювальних систем; різні аспекти програмування, в тому числі питання розроблення і створення будь-якого програмного забезпечення; інформаційні структури; мови програмування тощо.

При цьому обчислювальна техніка розв'язує задачі оптимізації апаратних засобів з точки зору їх побудування, енергоспоживання, надійності, вартості тощо.

Крім того, термін "**обчислювальна техніка"** застосовується для описування сукупності обчислювальних засобів, призначених для автоматизації розв'язання будь-яких задач керування процесами і збирання даних, таких як мікропроцесор, абонентські термінали, пристрої обміну даними тощо, а також окремі функціональні вузли цих засобів.

Електронна обчислювальна машина (ЕОМ), комп'ютер – комплекс технічних засобів, призначених для автоматизованого оброблення інформації в процесі розв'язання обчислювальних та інформаційних задач.

Під **обчислювальним пристроєм** зазвичай розуміють будь-який пристрій оброблення цифрової інформації: електронна обчислювальна машина (ЕОМ), мікропроцесор, персональний комп'ютер (ПК), мікропроцесорна система тощо.

Обчислювальна система (ОС) – це сукупність програм та технічних засобів, призначених для оброблення інформації.

Архітектура обчислювальної системи – це загальна логічна організація обчислювальної системи, яка визначає процес оброблення даних у ній та поєднує методи кодування даних, склад, призначення, принципи взаємодії технічних засобів і програмного забезпечення.

Процесор – це функціональний пристрій, що забезпечує конкретне застосування сукупності команд.

Мікропроцесор (МП) – це обробляючий та керуючий цифровий пристрій, виконаний за технологією великих інтегральних схем (ВІС), який під програмним керуванням здатний виконувати оброблення інформації, а саме

арифметичні та логічні операції, введення-виведення та зберігання інформації, а також приймати рішення.

За типом архітектури розрізняють МП з фоннеймановською архітектурою і МП з гарвардською архітектурою.

За типом побудови мови програмування розрізняють CISC-процесори (Complete Instruction Set Computing) з повним набором команд та RISC-процесори (Reduced Instruction Set Computing) – зі зменшеним набором команд.

Сучасні мікропроцесори мають ознаки як CISC, так і RISC-архітектури.

Мікропроцесорна система (МПС) – це багатофункційна програмно-керована система обробки інформації, яка складається з підсистеми центрального процесора, підсистеми пам'яті та підсистеми введення-виведення, об'єднаних інформаційними каналами. Мікропроцесорні системи будують на мікропроцесорних комплектах і поділяють на МПС керувальні, обчислювальні, контрольно-вимірювальні, збирання даних.

Комп'ютер сам по собі є також мікропроцесорною системою.

Різниця між обчислювальною системою та мікропроцесорною системою є тільки у масштабах розв'язуваних задач, кількості та складності обладнання.

Класичним варіантом ОС є багатокомп'ютерний або багатопроцесорний комплекс.

Вимоги до сучасних ОС – це, перш за все, здатність до інформаційного обслуговування користувачів, сервіс та якість цього обслуговування. Для суперкомп'ютерів, які будуються на основі багатопроцесорних систем, найбільш важливими вимогами є продуктивність та надійність.

Обчислювальні системи можуть будуватись на базі кількох комп'ютерів або на базі кількох процесорів.

У багатокомп'ютерних системах інформаційно комп'ютери взаємодіють один з одним по мережі, і кожен з них може працювати під керуванням своєї операційної системи, що знижує динамічні характеристики та надійність ОС.

Багатопроцесорні ОС працюють під керуванням однієї операційної системи, мають більш високу швидкодію та надійність.

Універсальні мікропроцесори призначені для застосування в обчислювальних системах та керувальних персональних комп'ютерах, робочих станціях, серверах, у суперкомп'ютерах. Основною характеристикою універсальних мікропроцесорів є наявність розвинених пристроїв для ефективної реалізації операцій з плаваючою точкою над 64-розрядними і більш довгими операндами та можливість підімкнення розвиненої системи введення-виведення інформації, яка забезпечує різноманітні види зовнішніх пристроїв. Вони призначені, в основному, для розв'язання науково-технічних, економічних, математичних, інформаційних та інших задач, які характеризуються складністю алгоритмів та великим обсягом даних, що обробляється.

Процесори цифрового оброблення сигналів (сигнальні процесори) – розраховані на оброблення у реальному часі цифрових потоків. Сучасні сигнальні процесори здатні виконувати цілочислові операції та операції з плаваючою точкою над 32-40-розрядними операндами.

Сигнальні процесори апаратно підтримують фільтрацію та згортання сигналів, обчислення кореляційної функції двох сигналів, пряме та обернене Фур'є-перетворення сигналу тощо.

Медійні та мультимедійні мікропроцесори забезпечують апаратну підтримку оброблення аудіосигналів, графічної інформації, відеозображень тощо. Вони застосовуються у мультимедіакомп'ютерах, приставках для ігор, побутовій техніці тощо.

Мікроконтролер є інтегрована на одному кришталі ВІС мікропроцесорна система, яка складається з одного або кількох мікропроцесорів, іноді з різною архітектурою та призначенням, підсистеми пам'яті та набору периферійних пристроїв. Мікроконтролери знаходять широке застосування у телекомунікаціях (комунікаційні мікроконтролери), засобах автоматизації, апаратурі зв'язку, контрольно-вимірювальній техніці тощо.

Мікропроцесорний комплект – це сукупність інтегральних мікросхем різного ступеня інтеграції: ВІС мікропроцесорів, ВІС пристроїв пам'яті, ВІС пристроїв введення-виведення, ВІС контролерів зовнішніх пристроїв, службових інтегральних схем – тактовий генератор, модулі, з яких складається інтерфейс з пам'яттю, контролери шин, арбітри шин тощо, які є сумісні за своїми електричними, інформаційними та конструктивними параметрами. Мікропроцесорні комплекти призначені, в основному, для побудови мікропроцесорних систем.

Контролером називається пристрій, часто вбудований, який призначений для керування технологічним пристроєм або процесом. Контролер будується на основі одного або кількох процесорів або мікроконтролера.

Трансп'ютери – це мікрокомп'ютери з власною внутрішньою пам'яттю та каналами (лінками) для підключення до інших трансп'ютерів з метою створення багатопроцесорних систем та паралельних обчислювальних систем.

Контрольні запитання:

1 З яких пристроїв можуть складатись ОС?

2 Які вимоги пред'являються до сучасних ОС?

3 Які задачі можуть вирішуватись за допомогою обчислювальної техніки?

4 Як можна класифікувати мікропроцесори за їх функціональним призначенням?

5 Як можна класифікувати МПС за їх функціональним призначенням?

6 Чим різняться між собою мікропроцесори, мікроконтролери та мікропроцесорні системи?

7 Що таке мікропроцесорний комплект?

8 Що таке трансп'ютер?

Контрольні запитання підвищеної складності:

1 Назвіть відомі Вам моделі ЕОМ; в яких роках вони були виготовлені?

2 Назвіть відомі Вам моделі ПК; в яких роках вони були виготовлені?

1.2 Принципи побудови та функціонування обчислювальних систем

Вхідний контроль:

1 Які задачі розв’язують обчислювальні системи?
2 У чому різниця між поняттями “кількість інформації” та “обсяг даних”?
3 Які задачі розв’язують інформаційні системи?

1.2.1 Архітектура обчислювальних систем

За архітектурою обчислювальні системи поділяються на **однорідні** та **неоднорідні**.

Однорідні обчислювальні системи побудовані на базі однотипних комп’ютерів або процесорів. Вони використовують стандартні набори технічних, програмних засобів, стандартні протоколи сполучення пристроїв.

Неоднорідні обчислювальні системи мають у своєму складі різні типи комп’ютерів або процесорів, і при побудові такої системи треба враховувати їхні різні технічні та функціональні характеристики, що ускладнює їх створення та обслуговування. Прикладом неоднорідної інформаційно-обчислювальної системи є мережа Інтернет.

Обчислювальні системи працюють у двох режимах – **оперативному** та **неоперативному**.

Оперативні системи працюють у реальному масштабі часу, в них реалізується оперативний режим обміну інформацією – відповіді на запитання надходять негайно.

У **неоперативних** системах можливий режим затриманої відповіді, коли результати запиту можна отримати з затримкою.

Розрізняють обчислювальні системи з **централізованим** та **децентралізованим керуванням**. У першому випадку керування виконує окремий комп’ютер або процесор, у другому – кожний компонент може брати керування на себе і вони є рівноправні.

Обчислювальні системи можуть бути розподіленими територіально, зосередженими, структурно однорівневими, тобто мати один загальний рівень обробки даних, та багаторівневими, ієрархічними структурами; у ієрархічних структурах комп’ютери або процесори розподілені за різними рівнями оброблення інформації і кожний з них може ініціалізуватись для виконання певних функцій.

За способом організації оброблення даних багатопроцесорні системи можуть бути **магістральними** (конвеєрними), **векторними** або **матричними**.

У **конвеєрних** багатопроцесорних системах кожен процесор одночасно виконує різні операції над послідовним потоком оброблюваних даних. За прийнятою класифікацією такі системи є системами з множинним потоком команд та поодиноким потоком даних (МКПД) – Multiple Instruction Single Data, MISD. Структура MISD показана на рис. 1.1.

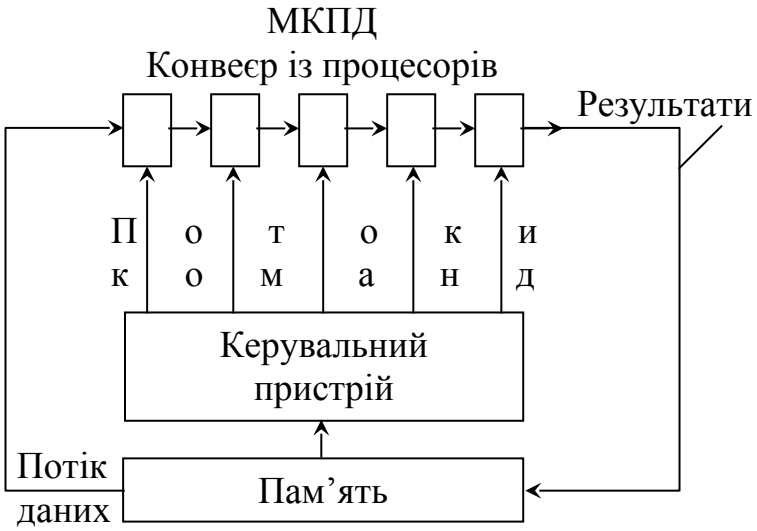


Рисунок 1.1 – Конвеєрна багатопроцесорна система
(множинний потік команд та поодинокий потік даних)

У **векторних** багатопроцесорних системах усі процесори одночасно виконують одну команду над різними даними – поодинокий потік команд з множинним потоком даних (ПКМД) – Single Instruction Multiple Data, SIMD. Структура SIMD показана на рис. 1.2. Принцип SIMD використовується також для підвищення продуктивності мікропроцесорів – **суперскалярні** (векторні) МП Pentium III, Pentium 4, Power PC тощо.

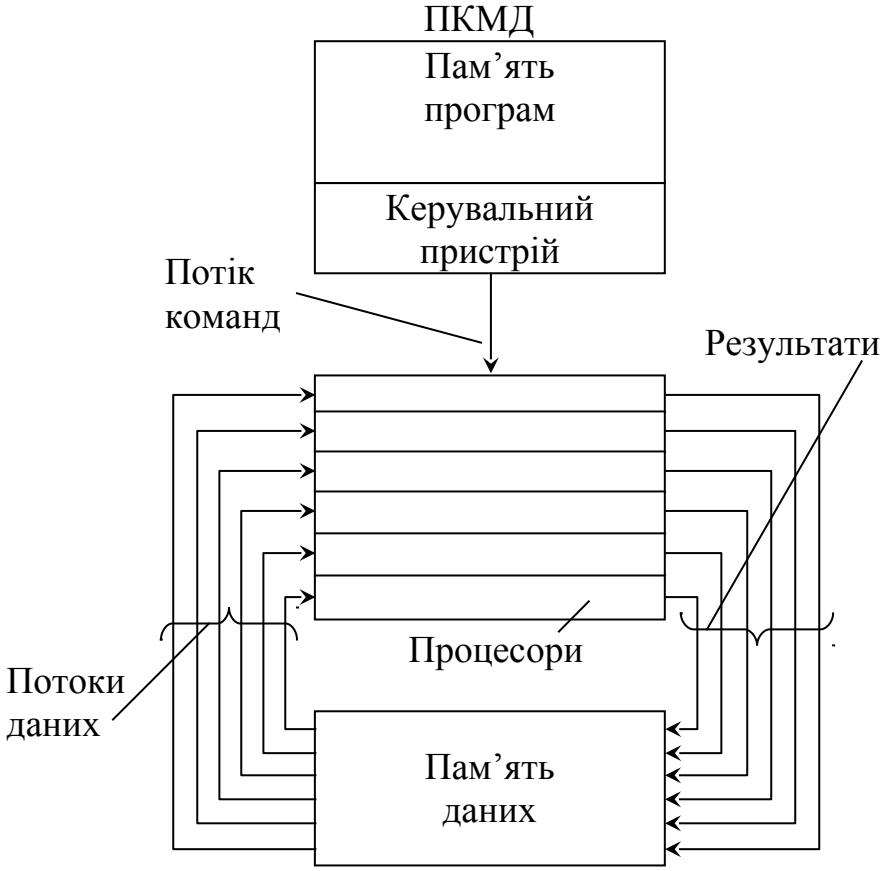


Рисунок 1.2 – Векторна багатопроцесорна система
(поодинокий потік команд та множинний потік даних)

У **матричних** багатопроцесорних системах кожний мікропроцесор одночасно виконує різні операції над послідовними потоками оброблюваних даних – множинний потік команд з множинним потоком даних (МКМД) – Multiple Instruction Multiple Data, MIMD. Структура MIMD показана на рис. 1.3.

Структура однопроцесорної системи (ПКПД) – Single Instruction Single Data, SISD показана на рис. 1.4.

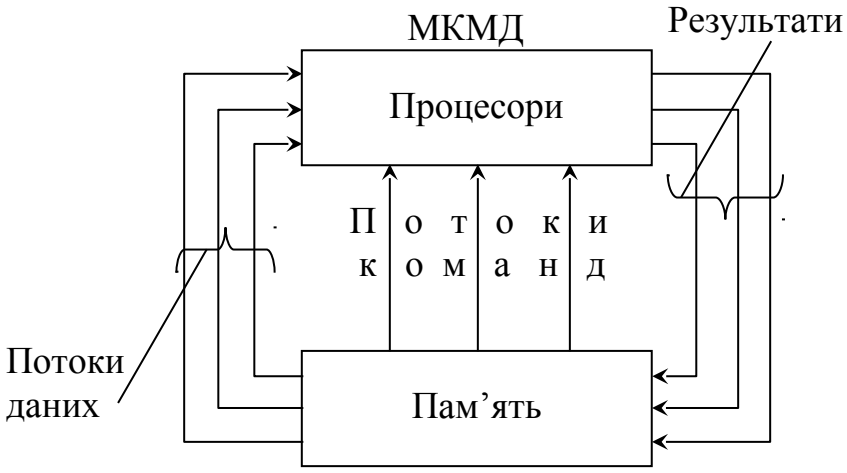


Рисунок 1.3 – Матрична багатопроцесорна система
(множинні потоки команд та даних)

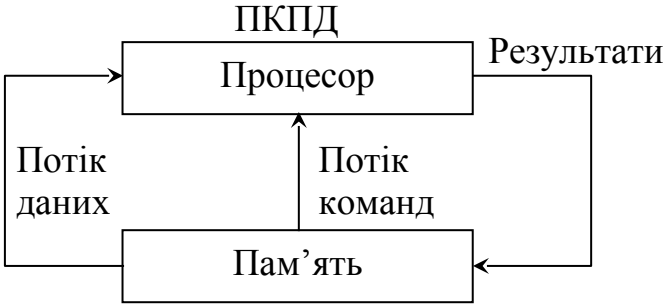


Рисунок 1.4 – Однопроцесорна система
(поодинокі потоки команд та даних)

Контрольні запитання:

1 Що є архітектурою обчислювальних систем?
2 Які режими обчислювальних систем Ви знаєте?
3 Які різновиди багатопроцесорних систем Ви знаєте?

Контрольні запитання підвищеної складності:

1 За яким принципом працює конвеєрна багатопроцесорна система?
2 За яким принципом працює векторна багатопроцесорна система?
3 За яким принципом працює матрична багатопроцесорна система?

1.2.2 Класифікація комп'ютерів (Для поглибленого вивчення)

Комп'ютери можна класифікувати за багатьма ознаками, що призводить іноді до різного тлумачення термінів.

За призначенням комп'ютери можна поділити на універсальні, проблемно-орієнтовані та спеціалізовані.

Універсальні комп'ютери використовують для розв'язання задач, які характеризуються складністю алгоритмів, великим обсягом оброблюваної інформації та вимагають високої продуктивності обчислювальних засобів. Це задачі математичні, інженерно-технічні, інформаційні, космічної галузі тощо.

Проблемно-орієнтовані комп`ютери призначені для керування телекомунікаційними пристроями, роботами, різними технологічними процесами і потребують меншої продуктивності та обчислювальних ресурсів. До них можна віднести робочі станції, які призначені для виконання графічних, інженерних, проектувальних, видавницьких робіт тощо.

Спеціалізовані комп'ютери призначені для керування простими технічними пристроями, процесами, використовуються для спрощення та узгодження роботи вузлів обчислювальних систем.

За розміром і обчислювальною потужністю комп'ютери можна поділити на суперкомп'ютери, великі, малі, мікрокомп'ютери.

Суперкомп'ютери призначені для розв'язання задач керування складними оборонними комплексами, моделювання екологічних систем, прогнозування погоди тощо.

Великі комп'ютери часто називають також майнфреймами. Вони призначені для розв'язання науково-технічних задач, роботи з великими базами даних, керування мережами та їхніми ресурсами.

Малі комп'ютери орієнтовані на використання у керувальних обчислювальних комплексах, системах автоматизованого проектування, моделювання та штучного інтелекту.

Мікрокомп'ютери можна поділити на багатокористувацькі, які оздоблені кількома відеотерміналами, та персональні комп'ютери, однокористувацькі, які задовольняють вимоги універсальності використання.

Структурна схема персонального комп'ютера показана на рис. 1.5.

Мікропроцесор є центральним блоком персонального комп'ютера. Він призначений для керування роботою усіх блоків ПК та виконання арифметичних і логічних операцій над даними.

До складу мікропроцесора входять:

1 Арифметично-логічний пристрій (АЛП), який призначений для виконання усіх арифметичних та логічних операцій над числовими та символьними даними. У деяких моделях ПК для прискорення виконання операцій до АЛП підключається додатковий математичний співпроцесор.

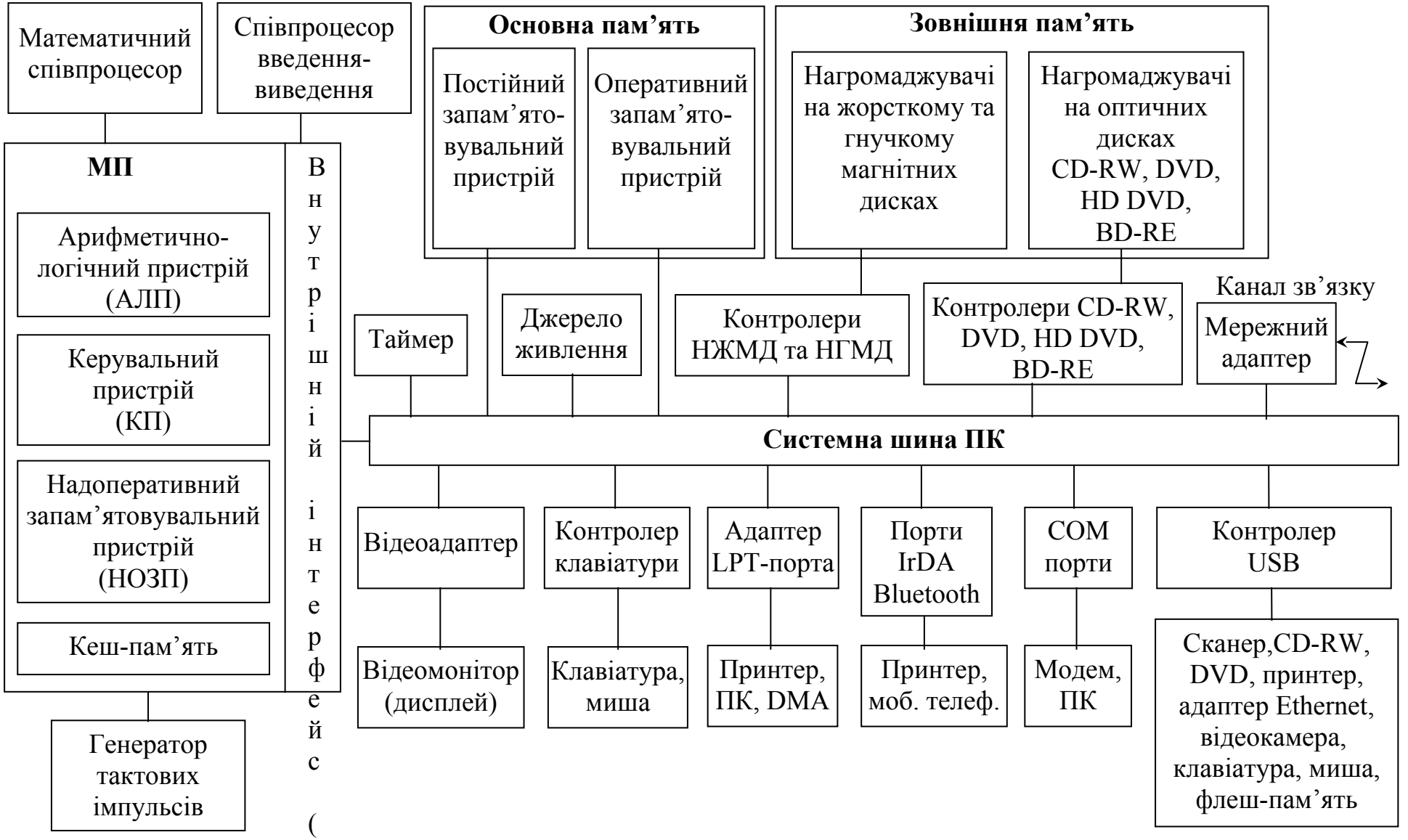


Рисунок 1.5 – Структурна схема ПК

2 Керувальний пристрій (КП) формує та подає на всі блоки ПК у визначені моменти часу певні сигнали керування, які обумовлені специфікою виконуваної операції і результатами попередніх операцій; формує адреси пам’яті і передає їх до відповідних пристроїв та вузлів комп’ютера; опорну послідовність імпульсів керувальний пристрій отримує від генератора тактових імпульсів.

3 Надоперативний запам’ятовувальний пристрій (НОЗП) призначений для короткочасного зберігання, записування та видавання інформації безпосередньо у найближчі такти роботи мікропроцесора. НОЗП побудований на регістрах.

4 Кеш-пам’ять команд та даних – це буферна пам’ять між мікропроцесором та оперативним запам’ятовувальним пристроєм або мікропроцесором та зовнішніми нагромаджувачами.

5 Інтерфейсна система мікропроцесора призначена для сполучення та зв’язку з іншими пристроями ПК; вона вміщує внутрішній інтерфейс мікропроцесора, буферні запам’ятовувальні регістри і схеми керування портами введення-виведення та системною шиною ПК.

6 Порт введення-виведення (I/O port) – це апаратура сполучення, яка дозволяє підключати до мікропроцесора інший пристрій ПК або периферійні пристрої.

7 Генератор тактових імпульсів генерує послідовність електричних імпульсів, частота яких визначає тактову частоту ПК. Частота генератора тактових імпульсів є одною з основних характеристик ПК і значною мірою визначає його продуктивність; кожна операція у ПК виконується за фіксовану кількість тактів.

Системна шина – це основна інтерфейсна система комп’ютера, яка забезпечує сполучення та зв’язок усіх його вузлів між собою.

Системна шина має у своєму складі:

− шину даних (ШД), яка створюється зі з’єднувальних ліній та схем сполучення для паралельного передавання усіх розрядів числового коду операнда;

− шину адреси (ША), яка вміщує з’єднувальні лінії та схеми сполучення для паралельного передавання коду адреси комірок основної пам’яті та порту введення-виведення зовнішнього пристрою;

− шину керування (ШК), яка складається зі з’єднувальних ліній та схем сполучення для передавання керувальних сигналів в усі блоки ПК;

− шину живлення, яка вміщує також з’єднувальні лінії та схеми сполучення для підключення блоків ПК до системи енергоживлення.

Системна шина забезпечує обмін даними між мікропроцесором та основною пам’яттю, між мікропроцесором та портами введення-виведення зовнішніх пристроїв, між основною пам’яттю та портами введення-виведення зовнішніх пристроїв у режимі прямого доступу до пам’яті.

Усі зовнішні пристрої ПК через їх порти введення-виведення підключаються до системної шини або безпосередньо, або через контролери (адаптери). Керування системною шиною ПК здійснюється частіш за все через

додаткову мікросхему – контролер шини, який формує основні сигнали керування.

Основна пам’ять призначена для керування та оперативного обміну інформацією з іншими блоками ПК. Основна пам’ять складається з постійного (ПЗП) та оперативного (ОЗП) запам’ятовувальних пристроїв:

− ПЗП (ROM – Read Only Memory) призначений для зберігання програмної та довідкової інформації, яка не змінюється; дозволяє тільки зчитувати інформацію, яка в ньому зберігається;

− ОЗП (RAM – Random Access Memory) динамічного та статичного типу призначений для оперативного записування, зберігання та зчитування інформації (програм та даних), які безпосередньо беруть участь в інформаційно-обчислювальному процесі.

Вимогами до оперативної пам’яті є її висока швидкодія та можливість доступу до будь-якої комірки пам’яті окремо. ОЗП є енергозалежною.

Крім основної пам’яті, у ПК є енергонезалежна пам’ять CMOS RAM (Complementary Metall-Oxide Semiconductor RAM), яка живиться від свого акумулятора; у ній зберігається інформація про апаратну конфігурацію ПК, яка перевіряється при кожному включенні ПК.

Зовнішня пам’ять є зовнішнім пристроєм ПК і використовується для зберігання усього програмного забезпечення ПК. Найбільш розповсюдженими видами зовнішньої пам’яті є нагромаджувачі на жорстких (НЖМД) та гнучких (НГМД) магнітних дисках, нагромаджувачі на оптичних дисках CD-ROM (Compact Disk Read Only Memory) та CD-RW (Re Writable CD), які дозволяють багаторазовий запис інформації.

У сучасних ПК для установлення програмного забезпечення, виконання певних додатків, наприклад, ігор, прослуховування музики, перегляду фільмів використовується DVD (Digital Versatile Disk) – універсальні цифрові диски, HD (High Density DVD) – диски DVD з високою щільністю запису на BD (Blu-ray Disk), які отримали таку назву від кольору проміня лазера, який зчитує інформацію з диску.

Флеш (Flash) – пам’ять з багаторазовим електричним стиранням та записуванням інформації.

Призначення зовнішніх нагромаджувачів – зберігання великих обсягів інформації, записування та видавання її по запиту в ОЗП. Зовнішня пам’ять є енергонезалежною.

Джерело живлення – це блок автономного та мережного енергоживлення.

Таймер – вбудований у ПК електронний годинник реального часу, який дозволяє автоматично видавати поточний момент часу (рік, місяць, години, хвилини, секунди та долі секунд). Таймер підключається до автономного джерела живлення – акумулятора, і при відключенні ПК від мережі продовжує працювати.

Зовнішні пристрої (ЗП) ПК забезпечують його взаємодію з зовнішнім інформаційним середовищем: користувачами, об’єктами керування, іншими комп’ютерами тощо.

До зовнішніх пристроїв відносяться:

- зовнішня пам’ять ПК;
- пристрої введення інформації;
- пристрої виведення інформації;
- діалогові засоби користувача;
- засоби зв’язку та телекомунікацій.

До пристроїв введення інформації відносяться:

- клавіатура;
- сканери;
- миша тощо.

До пристроїв виведення інформації відносяться:

- принтери;
- графобудувачі тощо.

LPT-порт (Line Printer Terminal) було введено у ПК для підключення принтера, але у розширеному режимі він може використовуватись як двоспрямований порт обміну даними і, як порт прямого доступу до пам’яті ПДП.

Контролер USB (Universal Serial Bus) дозволяє підключення до універсальної послідовної шини Flach-пам’яті, цифрових телевізійних камер, 10-Мбіт адаптерів Ethernet, багатопортових конверторів (USB to COM/USB-to-LBT), ноутбуків, клавіатур, цифрових фотоапаратів, CD-ROM, CD-RW, цифрових відеокамер тощо.

Деякі периферійні пристрої, найчастіше принтери, можуть мати бездротове з’єднання з ПК через інфрачервоний послідовний порт IrDA (Infrared Data Assotiation).

Радіоінтерфейс Bluetooth використовується для передавання інформації на невеликі відстані між ПК, їх периферійними пристроями, мобільними телефонами тощо.

Пристрої зв’язку та телекомунікацій використовуються для зв’язку з приладами та іншими засобами автоматизації (цифро-аналогові та аналого-цифрові перетворювачі, адаптери тощо), а також для підключення ПК до каналів зв’язку з метою обміну інформацією по мережі, у складі обчислювальних систем (мережні інтерфейсні плати та карти, модеми, мультиплексори передавання даних).

До діалогових засобів відносяться відеотермінал, керований відеокартою, та пристрої мовного введення-виведення інформації (мікрофони та акустичні системи), керовані звуковою картою. До засобів мультимедіа відносяться мікрофони, відеокамери, акустичні та відеосистеми тощо.

Математичний співпроцесор використовується для виконання операцій над двійковими числами з фіксованою та плаваючою точками, для обчислення трансцендентних, у тому числі тригонометричних функцій. Співпроцесор працює паралельно з основним мікропроцесором, що значно підвищує прискорення виконання операцій. Сучасні моделі мікропроцесорів часто інтегрують співпроцесор до своєї структури.

Співпроцесор введення-виведення працює паралельно з мікропроцесором і обслуговує кілька зовнішніх пристроїв (відеотермінал, принтер, НЖМД,

НГМД тощо) та звільняє процесор від оброблення процедур введення-виведення, у тому числі реалізує режим прямого доступу до пам'яті.

Контролер прямого доступу до пам'яті (DMA – Direct Memory Access) забезпечує обмін між зовнішніми пристроями та оперативною пам'яттю без участі мікропроцесора, що підвищує ефективність роботи ПК у цілому. Процесор під час обміну даними між зовнішніми пристроями та оперативною пам'яттю може обробляти інші дані або навіть вирішувати іншу задачу.

Контролер переривань обслуговує запити переривань від зовнішніх пристроїв за допомогою процедур переривань. Контролер приймає запит, визначає пріоритет цього запиту та визначає права конкретного зовнішнього пристрою на його обслуговування. Він посилає у мікропроцесор сигнал переривання та повідомляє його про номер або адресу обслуговуваного пристрою або про адресу першої команди підпрограми оброблення цього запиту. Мікропроцесор призупиняє виконування поточної програми та виконує підпрограму обслуговування переривання. Після завершення підпрограми мікропроцесор повертається до виконання перерваної програми.

Контролер переривань є програмований. Режим переривань використовується у ПК постійно, усі процедури введення-виведення виконуються за запитами зовнішніх пристроїв на переривання. Системний таймер здійснює перемикання задач, вирішуваних на ПК у багатозадачному режимі, кожні 2 мс.

Слід зазначити особливості термінології щодо комп'ютерів, об'єднаних в обчислювальні та інформаційні мережі.

Використання комп'ютерів у мережах визначається їх програмним забезпеченням та установленим додатковим устаткуванням.

Мережні комп'ютери – це спрощені мікрокомп'ютери, які забезпечують роботу у мережі та доступ до мережних ресурсів. Вони часто оздоблені відеотерміналом, клавіатурою, а іноді не мають навіть жорсткого диску. Такі комп'ютери можуть спеціалізуватись на виконанні певних робіт: захисту мережі від несанкціонованого доступу, перегляду мережних ресурсів, організації електронної пошти тощо.

Робочі станції у мережах – це персональні комп'ютери, які об'єднані у мережу і виступають як вузли цієї мережі.

Сервер – це багатокористувацький потужний мікрокомп'ютер, призначений для оброблення запитів від усіх робочих станцій мережі.

Проксі-сервер – це робоча станція, на якій установлено спеціалізоване програмне забезпечення для безпосереднього зв'язку локальних мереж з Інтернет.

Наведена термінологія складалась протягом кількох десятків років і відображує стани розвитку обчислювальних систем та мереж на базі електронних обчислювальних машин (ЕОМ). Розвиток потужних мікропроцесорів призвів до того, що в інформаційних та телекомунікаційних мережах як кінцеві вузли найчастіше використовуються персональні комп'ютери. Для розв'язування складних задач керування та моделювання актуальним є застосування великих та суперкомп'ютерів.

Найбільш перспективною технологією побудови великих та суперкомп'ютерів є кластерні обчислювальні системи – групи високоефективних процесорів, об'єднаних у кластери. Їх перевагою є можливість гнучкого регулювання необхідної потужності системи шляхом підключення до кластера звичайних серійних серверів за допомогою спеціальних апаратних та програмних інтерфейсів. Кластерізація дозволяє маніпулювати групою серверів як одною системою, забезпечувати доступ будь-якого сервера до будь-якого блоку оперативної та дискової пам'яті. Кластерні системи характеризуються спрощеним керуванням під операційними системами, наприклад, Windows 2000 Enterprise фірми Microsoft; її компонент Wolfpack забезпечує також функції діагностики збоїв та відновлення системи.

Створити високопродуктивні комп'ютери на одному мікропроцесорі неможливо через обмеження, зумовлені кінцевою швидкістю поширення електромагнітних хвиль (300 000 км/с), тому що час розповсюдження сигналу на відстань кілька міліметрів, яка складає розмір сторони МП при швидкодії 100 млрд. операцій за секунду, стає вже відповідним часу виконання однієї операції. Тому суперкомп'ютери створюються як високо паралельні системи.

У 2005 році Національна академія наук України створила в Інституті кібернетики ім. Глушкова (Київ) суперкомп'ютерний обчислювальний центр (СОЦ) на базі двох високопродуктивних кластерних систем СКІТ-1 та СКІТ-2 – 32-процесорного кластера на процесорах Intel Xeon та 64-процесорного – на процесорах Intel Itanium2.

Кластерна система СКІТ-1 (суперкомп'ютер для інформаційних технологій) – це 32-процесорний 16-вузловий кластер на основі 32-розрядних мікропроцесорів Intel Xeon, з піковою потужністю не менше 170 Гігафлопс (мільярдів операцій з плаваючою точкою за секунду) та можливістю підвищення продуктивності до 0,5-1 Терафлопс (трильйони операцій з плаваючою точкою за секунду). Кластер працює під керуванням головної операційної системи ALT Linux.

Система СКІТ-2 – це 64-процесорний 32-вузловий кластер на основі мікропроцесорів Intel Itanium2 з частотою 1,4 ГГц з розрядністю 64 біти і можливістю виконувати обчислення із 128 та 256-бітовою інформацією.

Пікова потужність кластера до 300 Гігафлопс з можливістю її підвищення до 2-2,5 Гігафлопс, підсистемою пам'яті 1 Гбайт і можливістю нарощування потужності до 10-15 Гбайт. Кластер працює під керуванням головної операційної системи Red Hat Enterprise.

Основними перевагами кластерних суперкомп'ютерних систем є:

- висока сумарна потужність;
- висока надійність системи;
- найкраще відношення потужність/вартість;
- можливість динамічного перерозподілу навантажень між серверами;
- легка масштабованість за рахунок підключення додаткових серверів;
- зручність керування та контролю роботи системи.

Суперкомп'ютери можуть мати також модифіковані структури – MMISD, паралельно-конвеєрна MISD-структура у суперкомп'ютері Ельбрус; MSIMD, паралельно-векторна модифікація, яка застосована у суперкомп'ютері Cray 2.

Слід зазначити, що багато ідей, втілених протягом десятків років у розроблених у різних країнах ЕОМ та комп'ютерах, знайшли застосування у сучасних мікропроцесорах.

Контрольні запитання:

1 З яких вузлів складається мікропроцесор?

2 З яких вузлів складається ПК?

3 З якою метою будуються кластерні суперкомп'ютери?

4 Які сучасні операційні системи керують кластерними системами?

5 Які новітні розробки українських вчених в області суперкомп'ютерів Ви знаєте?

6 Яку потужність має сучасний суперкомп'ютер, розроблений українськими вченими?

7 Які переваги кластерних суперкомп'ютерних систем Ви знаєте?

8 Які параметри комп'ютера слід враховувати при його виборі?

Контрольні запитання підвищеної складності:

1 На якій підставі може взаємодіяти персональний комп'ютер з периферійними пристроями через інфрачервоний послідовний порт?

2 На якій саме відстані може взаємодіяти мобільний телефон з ПК через радіо інтерфейс Bluetooth?

1.3 Принципи побудови та функціонування МПС

Вхідний контроль:

1 Перерахуйте підсистеми МПС.

2 Наведіть приклади об'єктів автоматичного контролю й управління.

3 Які пристрої введення та відображення інформації можуть використовуватись у керувальній МПС?

МПС будується за принципами "трьох М" – модульності, магістральності та мікропрограмованості. Модулем називається функціонально, електрично та конструктивно завершений цифровий пристрій, який призначено для виконання задач певного типу: процесорний модуль, модуль пам'яті тощо. Модульний підхід спрощує процес проектування МПС, орієнтованих на конкретні області використання, тобто найбільш ефективні, надійні, економічні.

Магістральний спосіб обміну інформацією в МПС реалізується у вигляді шинної організації, яка здійснює зв'язки між підсистемами МПС шинами (електричними лініями). Магістральність забезпечує регулярність структури МПС, можливість масштабування, змінення конфігурації, мінімізує кількість зв'язків між окремими пристроями. Зазвичай більшість універсальних мікропроцесорів забезпечують при побудові МПС тришинну організацію за

допомогою шин адреси (ША), шини даних (ШД) та шини керування (ШК), які утворюють системну шину.

Мікропрограмне керування може забезпечити найбільшу гнучкість у застосуванні МПС, але частіше використовують командний рівень керування через складність мікропрограмування.

Підсистеми МПС можуть складатися з кількох модулів мікропроцесорів, пам'яті, пристроїв введення-виведення.

Незважаючи на різноманіття МПС різного призначення, усі вони мають подібну структуру й однотипний склад устаткування. Їх характерною рисою є наявність розвиненої периферії або зовнішніх пристроїв: блоків датчиків, блоків керування, комутаторів, пристроїв введення та відображення інформації, наприклад, клавіатура, монітор. Зовнішні пристрої підключаються до системної шини МПС за допомогою її підсистеми введення-виведення за допомогою інтерфейса. На рис. 1.6 показана структурна схема МПС автоматичного контролю і керування технологічним процесом. До її складу входять блок датчиків $Д_1 \ldots Д_m$, власне МПС та блок керування $К_1 ... К_p$. Датчики слугують для вимірювання параметрів стану об'єкта автоматичного контроля та управління і можуть бути, як аналоговими ($Д_1 \ldots Д_n$), так і цифровими ($Д_{n+1} \ldots Д_m$). Аналогові та цифрові сигнали з датчиків підключаються до входів МПС за допомогою комутатора 1, після якого, у разі необхідності, ставиться АЦП.

МПС за заданими алгоритмами обробляє дані про стан процесу і видає цифрові керувальні сигнали через АЦП, якщо це потрібно, на блок керування $К_1 ... К_p$. Підключення керувальних сигналів до блоку керування здійснюється за допомогою комутатора 2. МПС може бути побудована на комп'ютерах та мікроконтролерах.

Блок керування може складатись з формувачів аналогових сигналів $К_1 ... К_n$ або цифрових сигналів $К_{n+1} ... К_p$ керування.

Керувальна МПС функціонує таким чином. Комутатор, керований МПС, опитує датчики за адресами, які задаються програмою, й інформація у цифровій формі надходить до МПС. На основі цієї інформації відповідно до програми роботи МПС формується модель стану об'єкта і видається інформація на пристрій відображення для оператора або на виконавчий механізм, наприклад, переключення ліній зв'язку тощо. Таймер формує відліки часу, мітки, які прив'язують процес до внутрішнього часу МПС. Датчик переривань забезпечує переривання поточної програми, наприклад, при виникненні аварійних ситуацій і перехід до програм їхнього оброблення.

Обмін даними між МПС та зовнішніми пристроями може реалізовуватись трьома способами: програмно керованим, за перериваннями та прямим доступом до пам'яті.

Програмно керований спосіб обміну даними ініціюється будь-яким процесором МПС за основною програмою, яка вміщує команди введення-виведення. Перед обміном процесор перевіряє готовність до роботи зовнішніх пристроїв. Цей спосіб є простий, але не забезпечує термінову реакцію МП на готовність зовнішніх пристроїв до обміну. Відповідно, такий програмно

керований спосіб використовується при роботі з повільно діючими різнорідними пристроями.

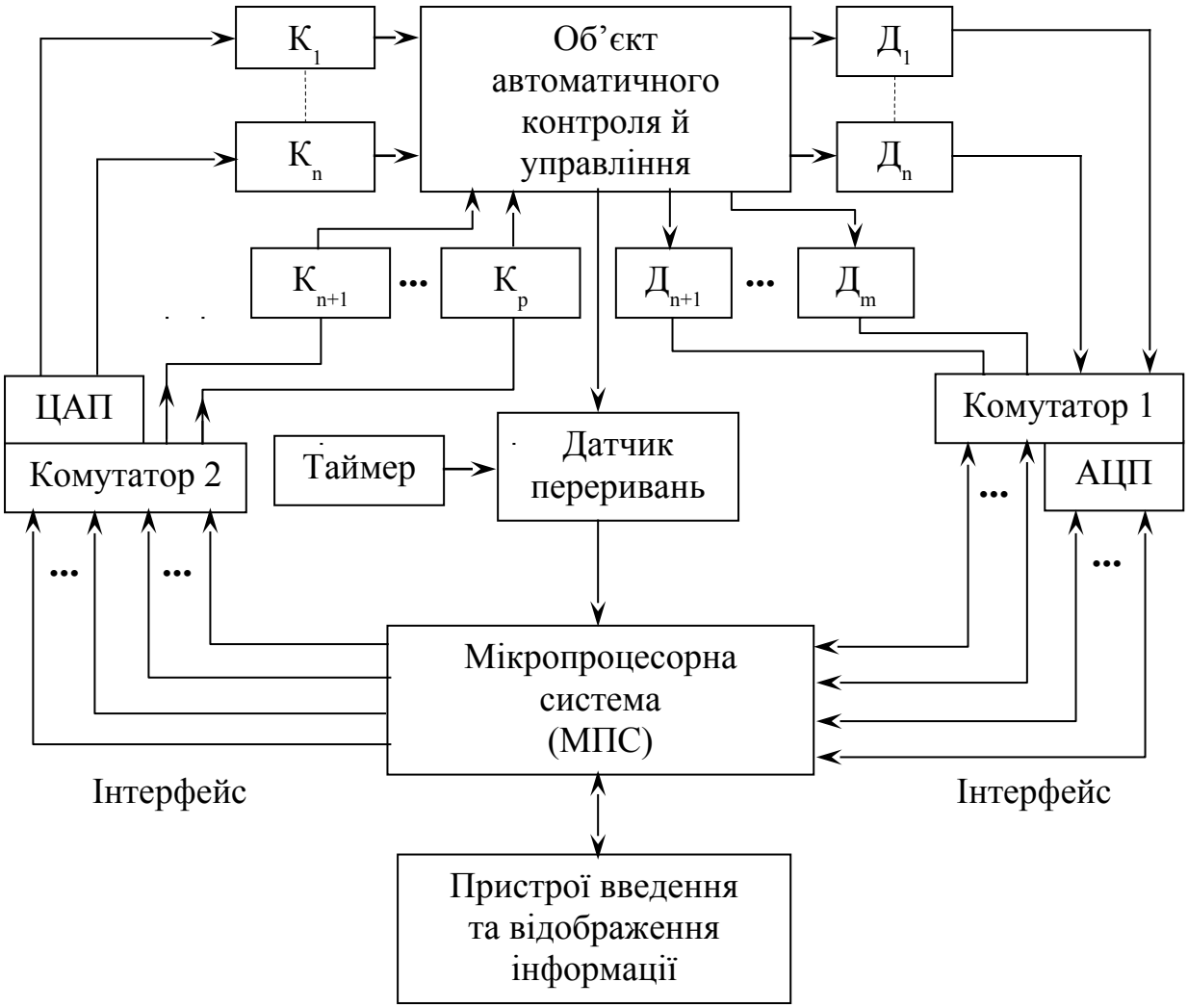


Рисунок 1.6 – Структурна схема МПС автоматичного контролю та управління технологічним процесом

При роботі з асинхронно, по відношенню до процесора, діючими пристроями доцільним є спосіб обміну за перериваннями. Зовнішній пристрій, коли він готовий до роботи, посилає сигнал INT – запит на переривання (від Interrupt – переривання) на відповідний вхід процесора. За цим сигналом, якщо переривання дозволені, процесор припиняє виконання поточної програми, посилає пристрою сигнал підтвердження INTA (від Interrupt Acknowledge – підтвердження переривання) і переходить до виконання підпрограми обслуговування переривання від даного пристрою. Метод переривань забезпечує швидку реакцію МПС на запити зовнішніх пристроїв, але також здійснює програмно керований обмін і потребує значної кількості команд на пересилання одного байта даних, оскільки обмін відбувається через процесор.

Найбільш високу швидкодію обміну забезпечує режим прямого доступу до пам'яті ПДП (DMA – Direct Memory Access) по каналу зовнішній пристрій-пам'ять. По запиту захоплення шин HOLD (від Hold – захоплення) від

зовнішнього пристрою процесор завершує поточний машинний цикл виконуваної команди і переводить системні шини адреси, даних і керування у стан високого опору і видає сигнал підтвердження захоплення HLDA. Процесор відключається від процесу обміну даними і обмін відбувається з максимальною для всіх учасників обміну швидкістю. Режим ПДП забезпечується за допомогою програмованих контролерів ПДП , які адресують комірки пам'яті, рахують біти та врегульовують конфліктні ситуації, які можуть виникнути при одночасній роботі кількох пристроїв у режимі ПДП.

Як правило, МПС є вузькоспеціалізованою після її розробки та впровадження. Розробка МПС здійснюється стосовно конкретної задачі (алгоритму), як в апаратній частині, так і у програмному забезпеченні. Робоча програма після налаштування МПС завантажується у ПЗП одноразово. Змінення реалізуючого алгоритму потребує зміни апаратної та програмної частини МПС, що не є ефективно.

Слід зазначити, що вбудовані у пристрої керування МПС, які розробляються на мікроконтролерах, допускають багаторазове програмування ПЗП з занесенням різних робочих програм, тобто перепрофілювання.

Контрольні запитання:

1 Яку роль відіграють АЦП та ЦАП у керувальній МПС?

2 З якою метою у керувальній МПС використовуються комутатори, датчики, керувальні блоки та таймер?

3 Назвіть основні етапи функціонування керувальної МПС.

4 Які вузли входять до складу підсистеми центрального процесорного елементу?

Контрольні запитання підвищеної складності:

1 Яку роль відіграє датчик переривань?

2 Наведіть приклад режиму переривань у ПК.

3 Чим на Ваш погляд відрізняється МПС збирання даних від керувальної МПС?

1.4 Функціонування обчислювального пристрою

Вхідний контроль:

1 Що таке обчислювальний пристрій?

2 Які обчислювальні пристрої Ви знаєте?

3 Які периферійні пристрої обчислювального пристрою Ви знаєте?

У 60-х роках минулого століття академік В. М. Глушков довів, що у будь-якому пристрої обробки цифрової інформації можна виділити операційний та керувальний блоки, що це є принцип декомпозиції обчислювального пристрою. Операційний блок складається з регістрів, суматорів та інших пристроїв, які приймають з запам'ятовувального пристрою, зберігають операнди, виконують над ними операції та видають результати операції у запам'ятовувальний

пристрій. У керувальний блок з операційного блоку надходять відомості про знак та інші особливості результату, наприклад, чи дорівнює він нулю тощо. Такі відомості називаються ознаки або прапорці F (від Flags) результату. На рис. 1.7 показано декомпозицію обчислювального пристрою.

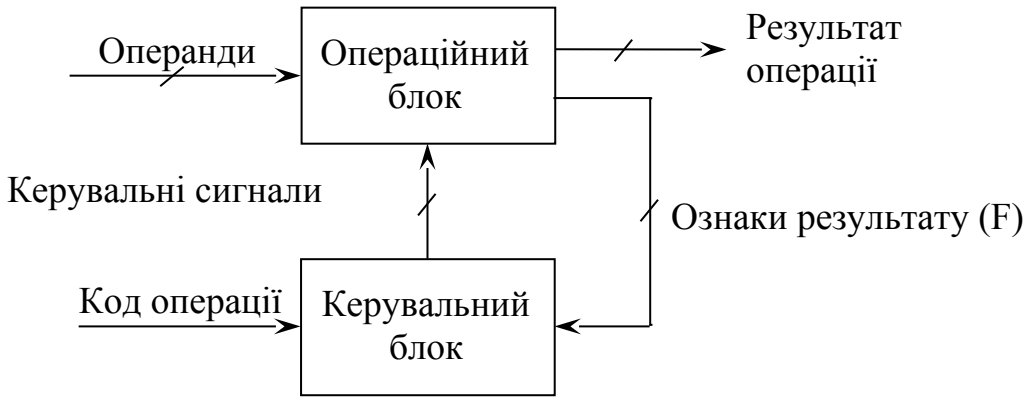


Рисунок 1.7 – Декомпозиція обчислювального пристрою

Процес функціонування пристрою оброблення цифрової інформації, зокрема обчислювального пристрою, складається із послідовності елементарних перетворень, які виконуються за інтервали часу, визначені частотою тактового генератора, такти. До елементарних операцій можуть відноситись зсув даних у регістрі, знаходження оберненого коду, пересилання операнда з одного регістра в інший тощо. Виконання елементарних операцій ініціюється поданням в операційний блок відповідних керувальних сигналів з керувального блока.

Елементарна функціональна операція, виконувана за один такт, називається **мікрооперацією**.

У деякі такти з керувального блоку можуть надходити кілька керувальних сигналів, які ініціюють виконання мікрокоманд у різних вузлах обчислювального пристрою. Сукупність одночасно виконуваних мікрооперацій називається **мікрокомандою**.

Послідовність керувальних сигналів визначається сигналами коду операції, які надходять після декодування у керувальний блок з пам’яті, і сигналами, які залежать від операндів та проміжних результатів обчислень.

Операційний блок задається його структурою, тобто складом вузлів та зв’язками між ними і виконуваним операційним блоком набору мікрооперацій.

Послідовність мікрокоманд, які забезпечують виконання даної операції, називається **мікропрограмою** даної операції.

Функціонування обчислювального пристрою може описуватись сукупністю мікропрограм, які в ньому реалізуються.

При створенні ВІС МП використовується ідея функціонування обчислювального пристрою. Спрощена схема обчислювального пристрою, або ЕОМ, ПК, МПС наведена на рис. 1.8. Вона складається з п’яти блоків: арифметично-логічного пристрою (АЛП) з надоперативним

запам’ятовувальним пристроєм (НОЗП), який складається з регістрів, керувального пристрою (КП), підсистеми пам’яті (ЗП) та підсистеми введення-виведення (ПВВ-ПВИВ) і побудована за апаратно-програмним принципом. Апаратна частина виконує обмежений набір простих операцій під керуванням програмного забезпечення. Обмін інформацією між підсистемами здійснюється за допомогою шини даних (ШД), шини адреси (ША) та шини керування (ШК).

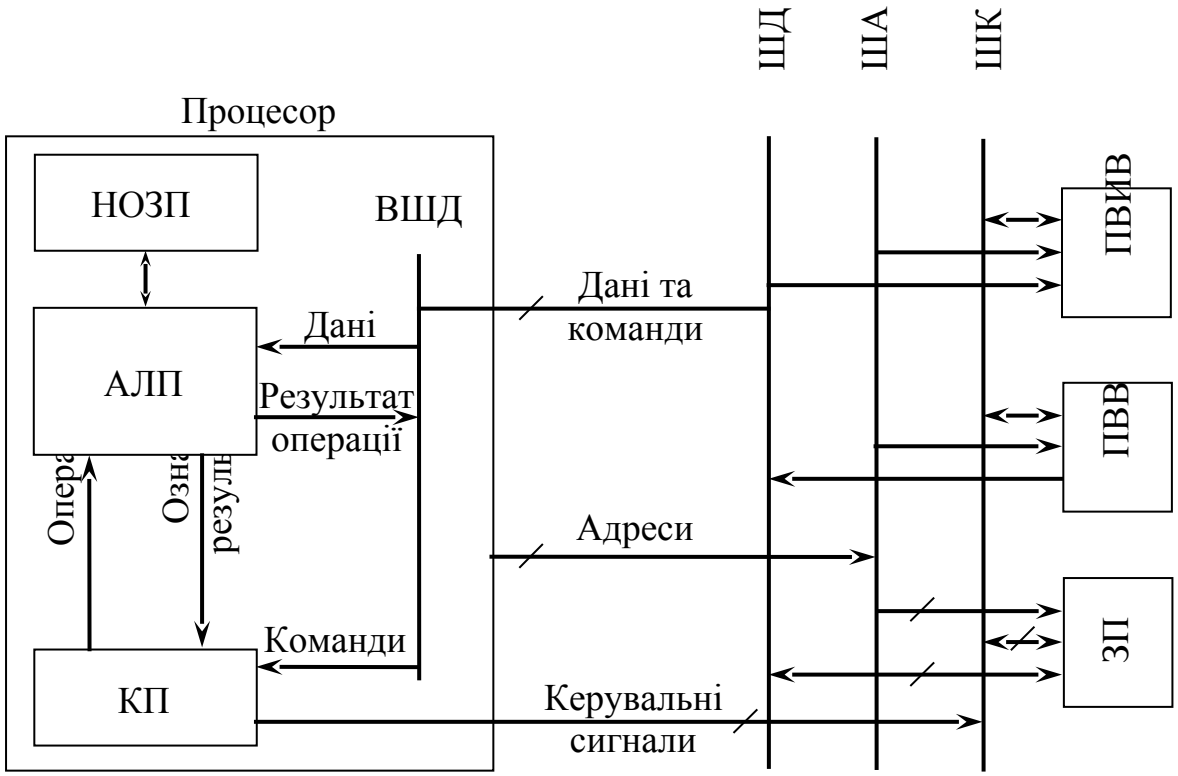


Рисунок 1.8 – Функціональна схема обчислювального пристрою

АЛП, НОЗП та КП входять до складу процесора і взаємодіють за допомогою внутрішньої шини даних (ВШД). Вхідні дані та програма задачі, яка розв’язується, надходять до запам’ятовувального пристрою через пристрій введення.

Виконувана програма складається зі списку команд (інструкцій), спрямованих на виконання серії послідовних дій (операцій). Кожна команда у свою чергу складається з операційної (код операції – КОП) та адресної частини, в якій вказуються самі операнди, або адреси операндів, над якими виконується задана операція. ЗП побудований за адресним принципом – кожна комірка пам’яті ЗП, в якій зберігаються дані або команди, окрім кеш-пам’яті, має постійний номер, адресу або ім’я, за якими можна однозначно до неї звертатись. Функціонування МПС починається з того, що КП центрального процесора видає адресу першої команди програми у ЗП, команда зчитується і направляється у КП. Тут вона декодується і на основі КОП команди виробляються сигнали Операція, які налаштовують АЛП на виконання заданої операції. За інформацією адресної частини команди з ЗП до АЛП надходять операнди. Після завершення операції її результат надходить для зберігання у

ЗП (Результат операції), а установлювані АЛП ознаки результату надходять до КП, що дає можливість вказувати розгалуження у програмі (приймати рішення). На лінійних ділянках програми далі виставляється адреса наступної команди, при розгалуженнях і циклах – залежно від реалізації заданих умов переходу.

Керувальний пристрій за результатами розшифровування команди виробляє паралельно-послідовну серію керувальних сигналів (КС) для керування як блоками самого процесора, так і підсистемами МПС.

Контрольні запитання:

1 Яка роль призначена апаратній частині і яка – програмній при вирішенні задачі на ЕОМ?

2 Як можна трактувати твердження, що процесор “приймає рішення”?

3 Який пристрій у складі процесора називається надоперативним і чому?

4 Як використовуються у виконуваній програмі ознаки результату виконаної команди?

Контрольні запитання підвищеної складності:

1 МПС призначена для оброблення сигналів, які надходять з цифрової ТВ камери. Який спосіб обміну даними між МП та камерою Ви б обрали?

2 Керувальна МПС призначена для керування системою комутації. Який спосіб обміну даними між МП та зовнішніми пристроями Ви б обрали?

2 ОПЕРАЦІЇ НАД ДАНИМИ В ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ

Вхідний контроль:

1 Які позиційні системи числення Ви знаєте? Їх особливості.

2 Як перевести число з однієї позиційної системи числення до іншої позиційної системи.

2.1 Подання даних в обчислювальних системах

Вхідний контроль:

1 Яке мінімальне та максимальне число без знака можна зберігати у 8-розрядному регістрі?

2 Яке мінімальне та максимальне число зі знаком можна зберігати у 8-розрядному регістрі?

3 Запишіть основу двійкової, десяткової та шістнадцяткової систем числення символами своєї ж системи?

4 У мережі Інтернет використовується єдиний для всіх країн світу універсальний 16-розрядний код (Unicod). Скільки символів можна відобразити за допомогою цього коду?

В обчислювальних системах виконуються необхідні дії по збиранню, зберіганню, обробленню і передаванню дискретної інформації, яка подана у вигляді потоку цифрових даних. Дані цифрового потоку можуть бути числами зі знаком або без нього, логічними змінними (масивами), адресами (номерами) будь-яких об’єктів і символами алфавітів, що використовуються. Усі ці види даних безпосередньо в обчислювальних системах представлені однаково – у вигляді двійкового коду, однак виконання арифметичних і логічних операцій виконується різними вузлами арифметико-логічного пристрою з використанням різних принципів. Так виконання арифметичних операцій виконується у позиційній двійковій системі числення, а виконання логічних операцій – за законами булевої алгебри.

Крім двійкової системи числення в обчислювальних системах також використовуються вісімкова і шістнадцяткова позиційні системи числення, двійково-десяткова система з кодованим поданням чисел.

Двійково-десяткова система числення з кодованим поданням чисел (**BCD-**система) **–** це система числення, в якій цифри десяткової системи числення кодуються за допомогою символів двійкової. Це система числення зі звичайною вагою двійкових розрядів – (8–4–2–1), в якій кожна десяткова цифра кодується двійковою тетрадою за допомогою двох символів двійкової системи. Слід зауважити, що двійково-десяткова система має певну надлишковість, тому що для кодування десяткових цифр використовується лише 10 комбінацій із 16 можливих у кожній тетраді. Це відрізняє її від інших позиційних систем числення, в яких надлишковості немає.

Використання цієї системи дозволяє зменшити програмні та апаратні витрати при перетворенні двійкових чисел на десяткові для виведення результатів на пристрої відображення і при введені інформації.

У комп'ютерах використовують упакований та розпакований формати подання двійково-десяткових даних.

Упакований формат передбачає подання двох цифр десяткової системи числення у двох тетрадах одного байта. Наприклад,

Байт

$$29_D \leftrightarrow \overbrace{\underbrace{0010}_{\text{Старша тетрада}}\ \underbrace{1001}_{\text{Молодша тетрада}}}^{\text{Байт}}{}_{BCD}$$

Таким чином, діапазон чисел у двійково-десятковій системі сягає від 00 до 99 і такі числа можливо використовувати при виконанні операцій додавання і віднімання з корекцією.

Розпакований формат передбачає подання десяткових чисел від 00 до 09 і використовується при виконанні операцій додавання, віднімання, множення і ділення з корекцією. У такому форматі при кодуванні вхідних даних у старшому розряді двійково-десяткового числа повинен бути лише нуль.

Крім двійково-десяткової системи числення зі звичайною вагою двійкових розрядів в обчислювальній техніці використовується двійково-десяткова система числення з надмірністю 3 (8–4–2–1 + 3)

Шістнадцяткова системи числення (Н-система) – система числення, яка дозволяє легко переходити від неї до двійкової і навпаки. Використовується для подання адреси пристроїв при написанні програм і для виводу результатів на екран. Так лістинг програми виводиться на екран у шістнадцятковій системі числення. Крім того, використання шістнадцяткової системи числення дозволяє зменшити обсяг тексту програми при її написанні мовою Асемблера.

Для подання алфавітно-цифрової інформації, що включає цифри, літери різних абеток, розділові знаки, математичні та інші символи, використовуються різні коди для оброблення такої інформації. Так у персональних комп'ютерах використовується код системи ASCII.

Система ASCII (American Standard Code for Information Interchange – Американській „стандартний код для обміну інформацією") є угодою за стандартом кодування символів, що схвалена Асоціацією стандартів США. Наявність такого коду спрощує обмін інформацією між різними пристроями комп'ютера. Фірма IBM запропонувала для використання у персональних комп'ютерах розширену версію системи ASCII, яка відрізняється від стандартної тим, що всі вісім біт символу використовуються для кодування інформації (у стандартній системі один біт призначався для перевірки правильності передавання інших семи бітів – перевірка на парність). Це дало змогу збільшити кількість символів на 128 і включити до їх складу символи національних алфавітів.

Стандартні коди ASCII можна розподілити на три групи:

– коди керування передаванням даних;
– коди керування форматом;
– коди друкованих символів.

Коди керування передаванням даних подають інформацію, що призначена для управління процесами обміну даними. Наприклад, код SOH (start of header – початок заголовка), код STX (start of text – початок тексту) і ETX (end of text – кінець тексту). Ці символи використовуються в комунікаційних протоколах для розмежування різних блоків даних, що передаються. Фірма IBM також модифікувала стандартне використання цих кодів, надавши їх друкованим графічним символам.

Коди управління форматом зображення використовуються для керування принтером або екраном монітору. Прикладами служать коди звукового сигналу, переходу на новий рядок, повернення каретки і прогону на початок сторінки. Коди керування передаванням даних і управління форматом зображення наведено у табл. 2.1. Друкарські символи стандартного набору наведено у табл. 2.2, а друкарські символи розширеного набору – у табл. 2.3.

Таблиця 2.1 – Коди керування передаванням даних і управління форматом зображення

Мнемо-ніка	Зображення при друкуванні	Призначення у системі ASCII	Код у системі числення	
			десятко-ва	шістнад-цяткова
NUL		Нульовий байт	0	0
SOH		Початок заголовку	1	1
STX		Початок тексту	2	2
ETX		Кінець тексту	3	3
EOT		Кінець передачі	4	4
ENQ		Запит	5	5
ACK		Підтвердження	6	6
BEL	Сигнал		7	7
BS	Повернення на один символ		8	8
HT	Табуляція		9	9
LF	Перехід на новий рядок		10	A
VT	Переміщення курсора в лівий верхній кут		11	B
FF	Прогін сторінки		12	C
CR	Повернення каретки		13	D

Закінчення табл. 2.1

Мнемо-ніка	Зображення при друкуванні	Призначення у системі ASCII	Код у системі числення	
			десятко ва	шістнад цяткова
SO		Нижній регістр	14	E
SI		Верхній регістр	15	F
DLE		Кінець сеансу зв’язку	16	10
DC1		Керування пристроєм 1	17	11
DC2		Керування пристроєм 2	18	12
DC3		Керування пристроєм 3	19	13
DC4		Керування пристроєм 4	20	14
NAK		Збій передавання даних	21	15
SYN		Синхронізація	22	16
ETB		Кінець блока передавання	23	17
CAN		Відміна	24	18
EM		Кінець носія даних	25	19
SUB		Підстановка	26	1A
ESC		Початок керування	27	1B
FS	Курсор праворуч	Розподілювач файлів	28	1C
GS	Курсор ліворуч	Розподілювач груп	29	1D
RS	Курсор вгору	Розподілювач записів	30	1E
US	Курсор вниз	Розподілювач полів	31	1F

Таблиця 2.2 – Друкарські символи стандартного набору

Символ	Код у системі числення		Символ	Код у системі числення		Символ	Код у системі числення	
	D	H		D	H		D	H
пропуск	32	20	@	64	40	`	96	60
!	33	21	A	65	41	a	97	61
“	34	22	B	66	42	b	98	62
#	35	23	C	67	43	c	99	63
$	36	24	D	68	44	d	100	64
%	37	25	E	69	45	e	101	65
&	38	26	F	70	46	f	102	66
‘	39	27	G	71	47	i	103	67
(	40	28	H	72	48	h	104	68
)	41	29	I	73	49	i	105	69
*	42	2A	J	74	4A	j	106	6A
+	43	2B	K	75	4B	k	107	6B
,	44	2C	L	76	4C	l	108	6C
–	45	2D	M	77	4D	m	109	6D
	46	2E	N	78	4E	n	110	6E
/	47	2F	0	79	4F	o	111	6F
0	48	30	P	80	50	p	112	70
1	49	31	Q	81	51	q	113	71
2	50	32	R	82	52	r	114	72
3	51	33	S	83	53	s	115	73
4	52	34	T	84	54	t	116	74
5	53	35	U	85	55	u	117	75
6	54	36	V	86	56	v	118	76
7	55	37	W	87	57	w	119	77
8	56	38	X	88	58	x	120	78
9	57	39	Y	89	59	y	121	79
:	58	3A	Z	90	5A	z	122	7A
;	59	3B	[	91	5B	{	123	7B
<	60	3C	\	92	5C	|	124	7C
=	61	3D	]	93	5D	}	125	7D
>	62	3E	^	94	5E	~	126	7E
?	63	3F	_	95	5F		127	7F

Таблиця 2.3 – Друкарські символи розширеного набору

Символ	Код у системі числення		Символ	Код у системі числення		Символ	Код у системі числення	
	D	H		D	H		D	H
А	128	80	а	160	A0	└	192	C0
Б	129	81	б	161	A1	┴	193	C1
В	130	82	в	162	A2	┬	194	C2
Г	131	83	г	163	A3	├	195	C3
Д	132	84	д	164	A4	─	196	C4
Е	133	85	е	165	A5	┼	197	C5
Ж	134	86	ж	166	A6	╞	198	C6
З	135	87	з	167	A7	╟	199	C7
И	136	88	и	168	A8	╚	200	C8
Й	137	89	й	169	A9	╔	201	C9
К	138	8A	к	170	AA	╩	202	CA
Л	139	8B	л	171	AB	╦	203	CB
М	140	8C	м	172	AC	╠	204	CC
Н	141	8D	н	173	AD	═	205	CD
О	142	8E	о	174	AE	╬	206	CT
П	143	8F	п	175	AF	╧	207	CF
Р	144	90	▒	176	B0	╨	208	D0
С	145	91	▒	177	B1	╤	209	D1
Т	146	92	▓	178	B2	╥	210	D2
У	147	93	│	179	B3	╙	211	D3
Ф	148	94	┘	180	B4	╘	212	D4
Х	149	95	┤	181	B5	╒	213	D5
Ц	150	96	╡	182	B6	╓	214	D6
Ч	151	97	╢	183	B7	╫	215	D7
Ш	152	98	╖	184	B8	╪	216	D8
Щ	153	99	╕	185	B9	┘	217	D9
Ъ	154	9A	╣	186	BA	┌	218	DA
Ы	155	9B	╗	187	BB	▉	219	DB
Ь	156	9C	╝	188	BC	▊	220	DC
Э	157	9D	╜	189	BD	▋	221	DD
Ю	158	9E	╛	190	BE	▌	222	DE
Я	159	9F	┐	191	BF	▍	223	DF

Закінчення табл. 2.3

Символ	Код у системі числення		Символ	Код у системі числення		Символ	Код у системі числення	
	D	H		D	H		D	H
р	224	E0	ы	235	EB	→	246	F6
с	225	E1	ь	236	EC	←	247	F7
т	226	E2	э	237	ED	↑	248	F8
у	227	E3	ю	238	EE	↓	249	F9
ф	228	E4	я	239	EF	÷	250	FA
х	229	E5	Ё	240	F0	±	251	FB
ц	230	E6	ё	241	F1	№	252	FC
ч	231	E7	′	242	F2	¤	253	FD
ш	232	E8	`	243	F3	*	254	FE
щ	233	E9	´	244	F4	пропуск	255	FF
ъ	234	EA	`	245	F5			

Двійкові числа у комірках пам’яті та регістрах мікропроцесора розміщуються таким чином, що для кожного біта (двійкового розряду) призначено окремий елемент пам’яті. Сукупність елементів, які утворюють комірку пам’яті, визначають довжину двійкового числа, називаються **розрядною сіткою**. Довжина розрядної сітки завжди обмежена і визначається конструктивними особливостями МПС. Тому значення чисел, що оброблюються, завжди обмежені. Ще більше обмежень стає при поданні дробових чисел.

Для подання дробових чисел в обчислювальній техніці використовуються дві форми:

- форма з фіксованою точкою;
- форма з плаваючою точкою.

При використанні форми з фіксованою точкою розрядна сітка конструктивно розподіляється на три частини: для подання знака, цілої і дробової частини числа. Використання такої форми пов’язано з труднощами при підготовці даних для оброблення, для забезпечення необхідного діапазону поданих даних. Також при обробленні даних трудно забезпечити необхідну точність оброблення (розрахунків).

Значно ефективніше використання форми з плаваючою точкою. Таке подання даних дозволяє розширити діапазон чисел і забезпечити необхідну точність подання результатів.

У цій формі дробове число подається у вигляді мантиси *M* і порядку *E (Exponent)*

$$D = \pm M \cdot B^{\pm E},$$

де *B* – основа системи числення, у нашому випадку *B = 2* – двійкова система числення.

Для однакового подання різних чисел мантису завжди нормалізують у межах

$$0{,}1 \le M < 1,$$

що виключає наявність нульового розряду системи числення після коми. Якщо у результаті обчислення, результат перестає бути нормалізованим, то перед його збереженням у пам'яті його необхідно нормалізувати, виконавши необхідний зсув мантиси і змінити значення порядку.

У формі з плаваючою точкою кількість розрядів порядку визначає діапазон оброблюваних чисел, а кількість розрядів мантиси – точність їх подання.

Контрольні запитання:

1 Для чого використовуються коди ASCII?

2 Які форми використовуються для подання дробових чисел?

3 Чим визначається точність подання чисел при використанні форми з плаваючою точкою?

Контрольні запитання підвищеної складності:

1 Подати двійкове число 111010101_B у шістнадцятковій системі числення.

2 Подати десяткові числа 06_D, 45_D, 60_D у двійково-десятковій системі числення в упакованому та розпакованому форматах.

3 Подайте число $12{,}25_D$ у вигляді з плаваючою точкою.

2.2 Подання даних у кодах

Вхідний контроль:

1 В чому полягає різниця між послідовними і паралельними кодами подання інформації?

2 Як записується дробове число у формі з фіксованою точкою?

3 Як записується дробове число у формі з плаваючою точкою?

З одного боку, в обчислювальній техніці існують два види кодів подання інформації – послідовний і паралельний, які обумовлюють порядок надходження даних для оброблення. Це пов'язано з конструктивними особливостями ліній передавання інформації, які бувають **двопроводовими** (телефонна лінія, радіорелейна лінія тощо) і **багатопроводовими** шинами. У двопроводовій лінії інформація передається побітно (у кожний момент часу на лінії є лише один біт) і код, що використовується, називається **послідовним**. Оброблення послідовного коду відбувається за декілька тактів (визначається розрядною сіткою), в міру надходження окремих бітів. Навпаки, у багатопроводовій шині у кожний момент часу інформація подана кількома розрядами числа, які можуть оброблятися одночасно, за один такт. Такий код називається **паралельним**.

З іншого боку, числа в обчислювальній техніці можуть подаватися у вигляді різних кодів у залежності від вимог до подання числа і необхідного його оброблення.

Отже, цілі беззнакові двійкові числа можуть бути подані натуральним кодом числа.

Натуральний код числа передбачає подання даних як цілих беззнакових двійкових чисел. Діапазон подання чисел у натуральному коді визначається довжиною розрядної сітки. При цьому максимальне число, що може бути подане у натуральному коді розрядної сітки становить 2^n–1, де n – кількість розрядів.

Необхідність виконання алгебраїчного додавання передбачає зображення числа сумісно зі своїм знаком, тому для подання таких чисел використовуються прямий, обернений і додатковий коди. Спосіб побудування цих кодів передбачає забезпечення таких вимог:

– запис алгебраїчного знака числа;

– подання від'ємних чисел за допомогою допоміжних додатних чисел, які відрізняються від відображення вихідних додатних чисел таким чином, щоб їх області зображення не збігалися;

– повна ідентичність алгоритмів виконання операцій над числами з однаковими і різними знаками, для забезпечення однотипності апаратного забезпечення для виконання цих операцій.

Прямий код (ПК) передбачає подання від'ємного числа таким чином, щоб у старшому розряді числа було показано його знак у вигляді 1, а в інших розрядах – його модуль. Старший розряд при цьому називають **знаковим**. Зображення додатного числа співпадає з його зображенням у натуральному коді, тому що знак + кодується у вигляді 0. При запису зручно знаковий розряд відокремлювати точкою після нього. Наприклад, $-98_D = 1.1100010_{ПК}$.

Діапазон представлення від'ємних чисел у прямому коді сягає від 0 до $2^{n-1} - 1$, а для додатних чисел – 2^{n-1}.

Операція додавання для чисел, поданих у прямому коді виконується по-різному, в залежності від їх знаків. Так, якщо числа мають однакові знаки, то числа додаються і сумі надається знак, який мають обидва числа. Якщо числа мають різні знаки, то спочатку знаходиться більше за модулем число, потім виконується віднімання від нього меншого і результату привласнюється знак більшого з них.

До недоліків використання прямого коду можна віднести:

– не забезпечується ідентичність алгоритмів виконання операцій над числами з однаковими і різними знаками, що приводить до збільшення кількості операцій для отримання результату;

– нуль може мати два значення: додатне число (у вигляді байта) – 0.0000000 і від'ємне число -1.000000, що також потребує виконання додаткових операцій при обчисленнях.

Для усунення цих недоліків і виконання операції віднімання (алгебраїчного додавання) в обчислювальній техніці використовуються **оберненний** і **доповнювальний** коди. Додатні числа у цих кодах подаються аналогічно прямому, а від'ємні обчислюються за певними алгоритмами.

Обернений код (ОК) від'ємного двійкового числа A, для певної розрядної сітки, обчислюється за виразом

$$A_{об} = N - |A|,$$

де N – значення найбільшого беззнакового числа, що можливо розташувати у певній розрядній сітці – значення N для десяткових дробів становить

$$N = 2 - 2^{-(n-1)},$$

а для цілих чисел

$$N = 2^n - 1.$$

У цих виразах n – кількість бітів розрядної сітки для подання числа. Діапазон подання чисел в оберненому коді, що можливо подати у розрядній сітці, такий самий, як і у прямому коді.

За визначенням обернений код від'ємного числа є доповненням його модуля до найбільшого беззнакового числа, яке можливо розмістити у розрядній сітці. Таким чином, взаємне перетворення прямого й оберненого кодів від'ємного числа виконується як операція порозрядної інверсії всіх розрядів числа крім знакового, в якому необхідно записати 1.

Наприклад,

$$-98_D = 1.1100010_{ПК} \Rightarrow \quad 1.0011101_{ОК}.$$

До переваг оберненого коду можливо віднести простий взаємозв'язок прямого й оберненого кодів, у результаті чого взаємне перетворення цих кодів є порозрядною операцією, що спрощує і прискорює її виконання.

Недоліками цього коду є необхідність урахування перенесення зі старшого розряду, яке виникає при виконанні додавання, і наявність двох значень для нуля: додатне – 0.0000000 та від'ємне – 1.1111111.

Доповнювальний код (ДК) від'ємного двійкового числа A для певної розрядної сітки обчислюється за виразом

$$A_{об} = K - |A|,$$

де K – значення ваги розряду, який знаходиться за старшим розрядом використовуваної розрядної сітки – значення K для десяткових дробів становить

$$K = 2,$$

а для цілих чисел

$$K = 2^n.$$

Діапазон подання чисел у доповнювальному коді для від'ємних чисел становить $0...2^{n-1}$, де n – кількість розрядів подання числа.

Доповнювальний код від'ємного числа легко отримати, додаючи одиницю молодшого розряду до оберненного коду цього числа.

Для попереднього прикладу

$$-98_D = 1.1100010_{ПК} \Rightarrow \quad 1.0011101_{ОК}$$
$$+ \quad \begin{array}{r} 1.0011101 \\ \underline{\qquad\qquad 1} \\ 1.0011110 \end{array}$$
$$1.0011101_{ОК} \Rightarrow \; 1.0011110_{ДК}$$

Для визначення прямого коду числа, яке подано у доповнювальному коді, віднімається одиниця із молодшого розряду, в результаті чого отримаємо оберненний код, а далі виконуємо інверсію всіх розрядів числа.

Використання доповнювального коду ліквідує недоліки оберненного коду: перенесення із старшого розряду втрачається і не враховується у подальших обчисленнях; нуль у доповнювальному коді – тільки додатне число.

Контрольні запитання:

1 Якими причинами обумовлене використання прямого оберненого і довнювального кодів?

2 Сформулюйте правило формування доповнювального кода.

3 Які недоліки існують при формуванні числа 0 в оберненому коді?

Контрольні запитання підвищеної складності:

1 Подати число +92 у вигляді восьмирозрядного (з урахуванням знака) прямого, зворотного та доповнювального кодів.

2 Подати число -121 у вигляді восьмирозрядного (з урахуванням знака) прямого, зворотного та доповнювального кодів.

2.3 Порозрядні операції над даними

Вхідний контроль:

1 Які правила виконання арифметичних операцій у позиційних системах числення Ви знаєте?

Виконання операції додавання у кодах дещо відрізняється від звичайного:

– розряди знаковий і розряди числа є рівноправними і перенесення у знаковий розряд необхідно враховувати і додавати до знакових розрядів;

– при виконанні операції додавання в ОК перенесення із знакового розряду необхідно додавати до молодшого розряду числа (операція циклічного перенесення);

– при виконанні операції додавання в ДК перенесення із знакового розряду ігнорується і відкидається.

Розглянемо операцію додавання цілих двійкових чисел зі знаком у кодах. Для подання у кодах необхідно користуватися восьмирозрядною сіткою (з урахуванням знака) у вигляді байта.

1 Для чисел з різними знаками

$$27_D - 30_D = 27_D + (-30_D) = 11011_B - 11110_B$$

представимо двійкові числа у кодах

ОК	ДК
$+ 11011_B \Rightarrow 0.0011011$	$+ 11011_B \Rightarrow 0.0011011$
$- 11110B \Rightarrow 1.1100001$	$- 11110B \Rightarrow 1.1100010$

Виконаємо операцію додавання у кодах

ОК	ДК
0.0011011	0.0011011
+ 1.1100001	+ 1.1100010
1.1111100	1.1111101

Результати отримано у відповідних кодах. Перетворимо їх на ПК і подамо у десятковій системі числення

$$1.1111100_{ОК} \Rightarrow 1.0000011_{ПК} \Rightarrow -3_D.$$

Результат при роботі з оберненним кодом правильний.

Для перевірки результату, поданого у ДК, його треба спочатку перевести в ОК, для чого від молодшого розряду необхідно відняти одиницю.

$$1.1111101_{ДК} - 1 = 1.1111100_{ОК} \Rightarrow 1.0000011_{ПК} \Rightarrow -3_D.$$

Результат також правильний.

Слід зазначити, що мікропроцесори виконують операцію додавання у доповнювальному коді, тому на етапі підготовки даних їх слід подати у вигляді ДК. Результат операції також подається в ДК. Для переведення його в ПК слід використовувати спеціальні програми, які виконують це перетворення.

2 Для двох інших чисел з різними знаками

$$-27_D + 30_D = -1011_B + 11110_B$$

представимо двійкові числа у кодах

ОК	ДК
$- 11011_B \Rightarrow 1.1100100$	$- 11011_B \Rightarrow 1.1100101$
$+ 11110_B \Rightarrow 0.0011110$	$+ 11110_B \Rightarrow 0.0011110$

Виконаємо операцію додавання у кодах

ОК	ДК
1.1100100	1.1100101
+ 0.0011110	+ 0.0011110
1 0.0000010	1 0.0000011
+ 1	
0.0000011	

При виконанні операції в ОК перенесення із знакового розряду було враховано в результаті, а при виконанні операції в ДК – відкинуто. Результат в обох випадках отримано у вигляді додатного числа $11_B = 3_D$, що є правильним.

При виконанні додавання даних, поданих у двійково-десятковій системі числення, виникає необхідність корегування результату в таких випадках:

– при формуванні суми у тетраді отримано число, яке не є цифрою десяткової системи числення, наприклад,

$$\begin{array}{rl} & 0000\ 1001_{BCD} \\ + & \underline{0000\ 0110_{BCD}} \\ & 0000\ 1111_{BCD} \end{array}$$

у молодшій тетраді отримано двійкове число 1111_B, яке неможливо подати у десятковій системі числення;

– при формуванні суми виникає перенесення до старшого розряду (старшої тетради), а значення результату в тетраді є невірне, наприклад,

$$\begin{array}{rl} & 0000\ 1001_{BCD} \\ + & \underline{0000\ 1000_{BCD}} \\ & 0001\ 0001_{BCD} \end{array}$$

у молодшій тетраді отримано двійкове число 0001В, замість 0111В, яке є правильним результатом.

В обох випадках корекція результату в тетраді буде виконуватися при використанні коду з надлишком 3 для обох складових, або при додаванні 6_D (0110_B) до результату. У першому випадку:

$$\begin{array}{rl} & 0000\ 1111_{BCD} \\ + & \underline{0000\ 0110_{BCD}} \\ & 0001\ 0101_{BCD} \end{array}$$

Контрольні запитання:

1 Чому може з’явитися необхідність корекції результату при виконанні арифметичних операцій над числами, які подані у двійково-десятковій системі числення?

2 Який код використовується для корекції результату арифметичної операції над числами, які подані у двійково-десятковій системі числення?

3 Чим відрізняється виконання арифметичних операції над числами зі знаком над числами, які подані в оберненому і доповнювальному кодах?

Контрольні запитання підвищеної складності:

1 Виконати операцію (+92 – 121) у доповнювальному коді. Отримати результат і перевести його у десяткову систему числення.

2 Виконати операцію (+92 – 121) в оберненному коді. Отримати результат і перевести його у десяткову систему числення.

3 Виконати операцію (+ 121 – 92) у доповнювальному коді. Отримати результат і перевести його у десяткову систему числення.

4 Виконати операцію (+ 121 – 92) в оберненному коді. Отримати результат і перевести його у десяткову систему числення.

5 Перевести число +8,73 у двійкову систему числення і подати у вигляді з плаваючою точкою (у вигляді *мантиси і порядку*). Мантису подати у вигляді округленого восьмирозрядного (з урахуванням знака) двійкового числа у прямому, зворотному та доповнювальному кодах.

6 Перевести число –5,63 у двійкову систему числення і подати у вигляді з плаваючою точкою (у вигляді *мантиси і порядку*). Мантису подати у вигляді округленого восьмирозрядного (з урахуванням знака) двійкового числа у прямому, зворотному та доповнювальному кодах.

7 Виконати операцію додавання чисел, отриманих у попередніх пунктах в оберненному коді.

8 Виконати операцію додавання (+ 8.73 –5,63) у доповнювальному коді.

9 Представити число 95 у двійково-десятковій системі числення.

10 Пояснити, чому при виконанні арифметичних операцій над числами, поданими у двійково-десятковій системі числення, необхідно використовувати код з залишком 3?

11 Як перевести десяткові числа 95 і 55 у двійково-десяткову систему числення? Виконати над отриманими числами операцію додавання. Скорегувати результат.

3 ЦИФРОВІ АВТОМАТИ

Вхідний контроль:

1 Які логічні операції виконуються в обчислювальній техніці?
2 Які властивості логічних елементів Ви знаєте?
3 Яка різниця між комбінаційними і послідовнісними схемами?

Перетворення інформації в обчислювальній техніці відбувається за допомогою електронних цифрових пристроїв (ЦП) – логічних схем, які будуються з логічних елементів і забезпечують виконання арифметичних і логічних операцій над сукупністю цифрових вхідних сигналів X_i, перетворюючи їх у сукупність вихідних сигналів Y_j. Розподіляють два класи таких пристроїв: **комбінаційні схеми** (КС) і **послідовнісні схеми** – цифрові автомати (ЦА).

В комбінаційних схемах результат перетворення – сукупність цифрових вихідних сигналів у будь-який момент часу t_n залежить лише від сукупності (комбінації) цифрових сигналів X_i, які надходять на її входи у даний момент часу і не залежать від стану схеми, який передував надходженню сигналів X_i. В таких схемах відсутні елементи пам'яті, тому сигнали, які діють у такий схемі, не зберігається. Такі ЦП ще називають цифровими автоматами без пам'яті (примітивними автоматами). До КС відносяться досить прості елементи, вузли та блоки мікропроцесорних систем: шифратори, дешифратори, мультиплексори, демультиплексори, комбінаційні суматори, перетворювачі кодів, схеми контролю тощо.

До **послідовністних схем** відносяться ЦП, в яких результат перетворення Y_j залежить не лише від комбінації цифрових сигналів X_i, які надходять на її входи у даний момент часу t_n, а й від послідовності попередніх цифрових станів входів і виходів схеми – внутрішніх станів Z_f. Для запам'ятовування попередніх станів така схема повинна мати елементи пам'яті. Подібні схеми називають **цифровими автоматами.** Можна стверджувати, що кількість внутрішніх станів і кількість елементарних запам'ятовувальних пристроїв в цифрових автоматах зв'язані між собою такою залежністю $Z=2^k$, де k – кількість запам'ятовувальних елементів.

Загальна теорія ЦА поділяється на **абстрактну** і **структурну**. Абстрактна теорія досліджує поведінку автомата відносно зовнішнього середовища, на рівні алгоритмів його роботи, не вирішуючи задач його побудови. Абстрактна теорія ЦА показує принципові відміни КС від ЦА для можливості подання дискретних процесів і обмеження, які є при цьому. Важливий висновок, який отримано у цих дослідженнях, полягає в тому, що в ЦА можливо реалізувати нескінченні регулярні події, на відміну від КС, де є можливість реалізувати лише скінченні події. Регулярними подіями вважаються такі події, які у скінченному алфавіті $X=\{x_1, x_2, \ldots x_n\}$ можливо отримати з елементарних слів алфавіту X при здійсненні над ними операцій диз'юнкції, перемноження та ітерації. Нерегулярні події реалізувати у ЦА неможливо, тому що це вимагає

нескінченного обсягу пам'яті. На практиці звичайно розглядаються ЦА, які мають обмежений обсяг пам'яті і називаються **скінченними ЦА.**

Структурна теорія досліджує проблеми побудування ЦА, кодування вхідних сигналів і реакції схеми на них. Використовуючи апарат булевої алгебри, структурна теорія надає важливі рекомендації щодо розробки схем пристроїв обчислювальної техніки.

У процесі проектування будь-якого ЦП доводиться вирішувати задачі аналізу або синтезу. Розв'язання задачі аналізу передбачає описання роботи принципової схеми ЦП в аналітичному вигляді, можливості його мінімізації й оцінку деяких параметрів готової структури ЦП.

Розв'язання задачі синтезу передбачає побудування структурної і електронної схеми пристрою відповідно опису алгоритму його роботи, для чого необхідно зробити вибір певних логічних елементів і передбачити їхнє з'єднання таким чином, щоб забезпечити виконання правил його функціонування, які подано в аналітичній формі. При розв'язанні цієї задачі необхідно зробити мінімізацію синтезованої схеми і виконати оцінку параметрів принципової схеми ЦП.

3.1 Визначення цифрових автоматів

Вхідний контроль:

1 Якими способами можливо показати алгоритми роботи цифрової схеми?

Абстрактний ЦА можна повністю описати за допомогою сукупності п'яти об'єктів:

- множини вхідних сигналів (вхідний алфавіт) – $X = \{x_0, x_1 \dots x_i\}$;
- множини вихідних сигналів (вихідний алфавіт) – $Y = \{y_0, y_1 \dots y_j\}$;
- множини стійких станів ЦА (алфавіт станів ЦА) – $Z = \{z_0, z_1 \dots z_f\}$;
- функції переходів – φ, яка визначає стан ЦА Z в залежності від вхідного сигналу і попереднього стану;
- функції виходів – λ, яка визначає залежність вихідного сигналу від вхідного і попереднього станів.

В залежності від способу формування функції виходів абстрактні ЦА поділяються на **цифрові автомати Мілі** і **цифрові автомати Мура**.

Якщо закон функціонування ЦА можна описати системою наступних рівнянь:

$$\begin{cases} y(t+1) = \lambda\{z(t), x(t)\} & \text{– вихідний сигнал ЦА після переключення;} \\ z(t+1) = \varphi\{z(t), x(t)\} & \text{– стан ЦА після переключення,} \end{cases}$$

де $t = 0, 1, 2, \dots$, то такий ЦА називається автоматом Мілі. Отже вихідний сигнал після переключення залежить від стану $z(t)$, в якому ЦП знаходився у момент надходження сигналу і вхідного сигналу $x(t)$.

В обчислювальній техніці широко використовуються пристрої, які можуть бути описані як ЦА Мура, який описується іншою системою рівнянь:

$$\begin{cases} y(t+1) = \lambda\{x(t)\} & \text{– вихідний сигнал ЦА після переключення ;} \\ z(t+1) = \varphi\{z(t), x(t)\} & \text{– стан ЦА після переключення,} \end{cases}$$

де $t = 0, 1, 2, ...$

Звідси вихідний сигнал такого ЦА не залежить від стану автомата, а повністю визначений вхідним сигналом.

Абстрактний ЦА, в якого виділено спеціальний початковий стан z_0 називається **ініціальним**. Зазвичай, всі вихідні сигнали у такому початковому стані дорівнюють нулю. Виділення такого стану обумовлено вимогами практичного використання пристроїв, в яких він є обов'язковим для правильного функціонування всієї системи. Таким чином, описати роботу ініціальних ЦА можливо так: якщо на вхід автомата Мілі або Мура, який знаходиться у початковому стані, подавати послідовність символів вхідного алфавіту $x(0)$, $x(1)$, $x(2)$, ..., то на виході отримаємо послідовність символів вихідного алфавіту. Тому на рівні абстрактної теорії функціонування ЦА описується як перетворення вхідного алфавіту у вихідний.

Якщо у ЦА функція переходів і функція виходів визначені для всіх пар X_i і Z_f, то він називається повністю визначеним. ЦА, для якого функція переходів або функція виходів, або обидві разом не визначені для всіх пар X_i і Z_f, називається **частково визначеним**.

Реальні ЦА завжди скінченні, тому що множина вхідних і вихідних сигналів, а також множина внутрішніх станів обмежені.

Функціонування ЦА відбувається у певні часові інтервали – такти. Залежно від способу визначення такту ЦА поділяються на **синхронні** і **асинхронні**.

Синхронні ЦА змінюють свій внутрішній стан у суворо визначені моменти часу, які завдаються за допомогою спеціального пристрою – генератора тактових імпульсів (ГТІ), який формує послідовність імпульсів спеціального сигналу синхронізації. При цьому тривалість тактового інтервалу визначена і залишається постійною протягом усього часу роботи схеми.

Асинхронні ЦА змінюють свій стан у довільні моменти часу, які визначаються моментом надходження (зміни) вхідних сигналів. При цьому, тривалість тактового інтервалу точно не визначена і дорівнює часу між сусідніми змінами вхідних сигналів. Зміна вхідних сигналів може відбуватися лише після завершення поточного такта.

Структурна схема ЦА у загальному виді показана на рис. 3.1.

До цієї схеми входять: КС1 – комбінаційна схема 1, яка формує сигнали запису інформації (вхідний алфавіт) у запам'ятовувальний пристрій – ЗП, який може бути побудований з тригерів, елементів затримки сигналів та інших пристроїв, що здатні запам'ятовувати інформацію. Цей пристрій фіксує внутрішній стан (алфавіт станів) ЦА і зберігає його протягом такту. КС2 – комбінаційна схема 2, яка формує вихідні сигнали (вихідний алфавіт) ЦА.

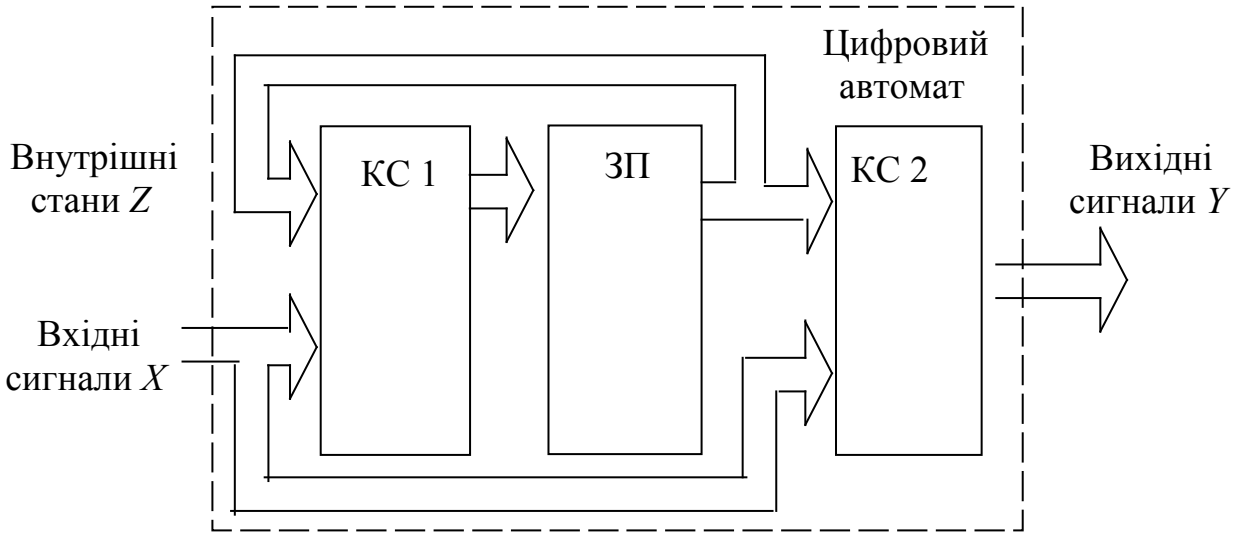


Рисунок 3.1 – Узагальнена структурна схема ЦА

Слід зазначити, що як вхідні сигнали також використовуються всі або будь-яка частина сигналів внутрішніх станів ЦА, які по ланцюгах зворотного зв'язку надходять на входи КС1 з виходів ЗП. На входи КС2 можуть надходити вхідні сигнали або деякі з них. Наявність таких зв'язків необхідна для спрощення схем КС1 і КС2.

Описування алгоритму роботи абстрактного ЦА можна задати трьома способами: **матричним**, **графічним** і **аналітичним**.

При використанні матричного способу алгоритм роботи ЦА подається у вигляді таблиці переходів і таблиці виходів, або у вигляді матриці з'єднань.

Таблиця переходів визначає функцію переходів ЦА – φ, а таблиця виходів – функцію виходів λ. Таблиця переходів для деякого конкретного ЦА наведена у табл. 3.1. Вона будується таким чином, що стовпці цієї таблиці визначають символи станів цього автомата (вважаємо, що кількість бітів інформаційних сигналів відповідних кожному стану дорівнює 4), і вхідного алфавіту, символи якого переключають ЦА у наступні стани (вважаємо, що в описуваному ЦА вхідний алфавіт складається з чотирирозрядних двійкових чисел). Рядки таблиці завдають символи алфавіту станів у порядку їх появи. Вважаємо, що цей ЦА є ініціальним і після проходження циклу перемикань ЦА повертається у початковий стан Z_0. Тому що ЦА описується лише чотирма станами, то він є частково визначеним (усього в цього ЦА може бути $Z=2^4 = 16$ стійких станів).

Таблиця 3.1 – Таблиця переходів ЦА

Стани ЦА, Z	Побітне значення алфавіту станів ЦА				Побітне значення алфавіту вхідних сигналів			
	z_3	z_2	z_1	z_0	x_3	x_2	x_1	x_0
Z_0	0	0	0	0	0	1	0	1
Z_1	0	1	0	1	0	1	1	1
Z_2	0	1	1	1	1	0	0	1
Z_3	1	0	0	1	1	0	1	1
Z_4	1	0	1	1	–	–	–	–

Таблиця виходів ЦА визначає вихідні сигнали у кожному зі стійких станів. Таблиця виходів ЦА, який описано табл. 3.1, подана в табл. 3.2.

Таблиця 3.2 – Таблиця виходів ЦА

Стани ЦА, Z	Побітне значення алфавіту станів ЦА				Вихід-ний алфавіт	Побітне значення алфавіту вихідних сигналів								
	z_3	z_2	z_1	z_0		y_8	y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0
Z_0	0	0	0	0	Y_0	0	0	0	0	0	0	0	0	0
Z_1	0	1	0	1	Y_1	0	0	1	1	0	0	0	1	1
Z_2	0	1	1	1	Y_2	0	0	1	1	0	1	0	1	1
Z_3	1	0	0	1	Y_3	0	1	0	0	1	0	1	1	0
Z_4	1	0	1	1	Y_4	1	1	1	0	1	0	1	1	1

Розглянуті таблиці можливо об’єднати у одну, яка має назву **відзначеної таблиці переходів**. Вид такої таблиці для розглядуванного ЦА показано в табл. 3.3.

У кожній клітинці цієї таблиці показано стан ЦА після переключення з поточного стану і сигнал , який формується на його виходах у поточному стані.

Риски у відповідних клітинках відповідають неможливим комбінаціям сигналів і станів.

Матриця з’єднань являє собою таблицю у вигляді квадратної матриці, кількість рядків і стовпчиків якої дорівнює кількості стабільних станів ЦА. Кожний рядок (стовпчик) відповідає стійкому стану, у якому може знаходитися ЦА. У кожній клітинці матриці показано символ (символи) вхідного алфавіту,

Таблиця 3.3 – Відзначена таблиця переходів

Стани ЦА, Z	Алфавіт вхідних сигналів				
	$X(4)$	$X(3)$	$X(2)$	$X(1)$	$X(0)$
$Z(0)$	– / –	– / –	– / –	– / –	Z_1 / Y_0
$Z(1)$	– / –	– / –	– / –	Z_2 / Y_1	– / –
$Z(2)$	– / –	– / –	Z_3 / Y_2	– / –	– / –
$Z(3)$	– / –	Z_4 / Y_3	– / –	– / –	– / –
$Z(4)$	Z_0 / Y_4	– / –	– / –	– / –	– / –

під впливом яких ЦА переключається у стан, який відповідає номеру стовпчика зі стану з номером рядка. Набір символів вхідного алфавіту з'єднується за допомогою відповідних логічних операцій.

Крім того, у кожній клітинці у дужках показано вихідний сигнал, який буде сформовано після переключення.

Матриця з'єднань для ЦА, який ми описуємо, показана на рис. 3.2.

	Z_0	Z_1	Z_2	Z_3	Z_4
Z_0	X_1 (Y_0)	–	–	–	–
Z_1	–	X_2 (Y_1)	–	–	–
Z_2	–	–	X_3 (Y_2)	–	–
Z_3	–	–	–	X_4(Y_3)	–
Z_4	–	–	–	–	X_0(Y_4)

Рисунок 3.2 – Матриця з'єднань цифрового автомата

При графічному описуванні функціонування робота ЦА описується орієнтованим графом, вершини якого відповідають стійким станам, а переходи зі стана в стан показують дугами між відповідними вершинами. Кожній дузі відповідає деяка літера вхідного алфавіту, яка викликає переключення, і відповідна літера вихідного алфавіту. Вид спрямованого графа, що відповідає алгоритму роботи ЦА, який описано вище, подано на рис. 3.3.

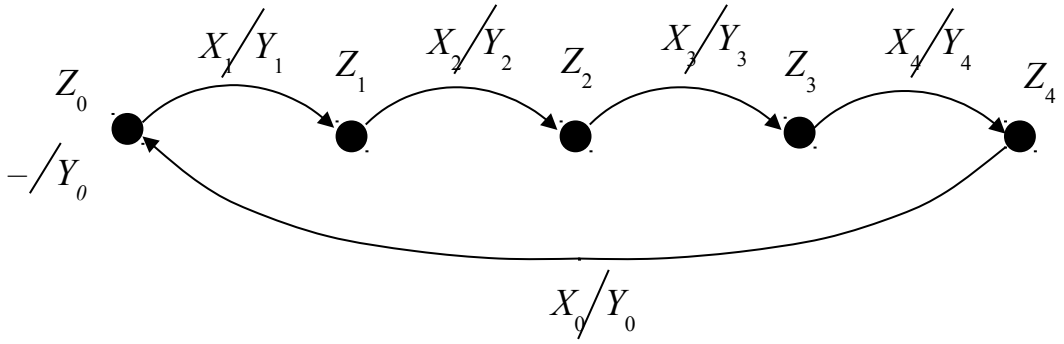


Рисунок 3.3 – Спрямований граф роботи цифрового автомату

При аналітичному описуванні роботи ЦА алгоритм його роботи задається у вигляді логічних функцій, що відповідають залежностям між різними характеристиками ЦА, які були описані вище.

При розробленні принципової схеми ЦА вигляд і склад запам'ятовувального пристрою, як правило визначено, тому задача розробки ЦА зводиться до задачі проектування цифрових пристроїв – комбінаційних схем КС 1 і КС 2, які представляють собою логічні схеми.

ЦА широко використовуються в обчислювальній техніці і у галузі зв'язку, як перетворювач інформації, і для генерування сигналів керування певними об'єктами в залежності від вхідних сигналів. При цьому в залежності від інформації, що надійшла, буде виконуватись один з багатьох можливих

алгоритмів. Стосовно до обчислювальної техніки такий алгоритм характеризує роботу і мікропроцесорної системи, і мікропроцесора, і їхніх вузлів.

Так, мікропроцесор, у самому широкому розумінні, складається з двох вузлів – **операційного блока** і **пристрою керування**. Пристрій керування забезпечує виконання команд програми і координацію дій складових частин операційного блока. Він генерує послідовність сигналів для виконання команди. Ця послідовність формується пристроєм керування у результаті надходження на його входи коду команди. Кількість сигналів у послідовності, їх чергування в часі і кількість символів у кожному із сигналів визначаються кодом команди, яка надходить на пристрій керування. Приклад побудування такого ЦА буде описано нижче.

У системах електрозв'язку ЦА використовуються: для керування пристроями і системами, для побудування різних вузлів і пристроїв зв'язку, для формування і розподілення сигналів у системах з часовим розподілом, для формування складних послідовностей вимірювальних сигналів і збереження результатів випробувань тощо.

Контрольні запитання:

1 Яка різниця між ЦА Мілі та Мура?

2 Які структурні елементи входять до схеми ЦА?.

3 Скільки стійких станів має ЦА, запам'ятовувальний пристрій якого побудовано на базі 5-розрядного паралельного регістра?

Контрольні запитання підвищеної складності:

1 Наведіть таблицю переходів частково визначеного ініціального ЦА, запам'ятовувальний пристрій якого побудовано на базі 3-розрядного паралельного регістра, якщо він має 4 стійкі стани і вихідний алфавіт якого складається із 4 символів (символи вихідного алфавіту і номери стійких станів вибрати самостійно).

3.2 Синтез логічних схем

Вхідний контроль:

1 Які логічні функції Ви знаєте?

2 Які способи мінімізації логічних функцій Ви знаєте?

Правила функціонування логічних схем (ЛС) можуть задаватися у різний спосіб:

- словесне описування;
- за допомогою таблиць істинності;
- за допомогою часових діаграм;
- аналітичний – за допомогою логічних (булевих) виразів;
- у вигляді координатних діаграм.

Усі ці способи описують один і той самий процес, тому по одному з них легко уявити функціонування ЛС і перейти від одного виду описування до іншого.

Слід зазначити, що для спрощення роз'язання задачі синтезу загальна ЛС поділяється на декілька схем, кожна з яких має один вихід (із сукупності вихідних сигналів Y_j) і стільки входів, скільки входить у сукупність вхідних сигналів X_i, необхідних для формування Y_j. Для більшого спрощення дозволяється подальше роздрібнення схеми таким чином, щоб зменшити кількість вхідних сигналів X. Після закінчення синтезу схем усі ці дрібні схеми об'єднують в одну загальну схему ЦП. Розв'язання задачі синтезу виконується за декілька етапів.

На першому етапі відбувається описування функціонування ЛС, як правило, у вигляді таблиці істинності, яка описує кожен із стійких станів ЦП, як співвідношення вхідних і вихідного сигналів і кожен рядок таблиці відповідає певному часовому інтервалу роботи. Описування виконується переважно для схем, які мають один вихід Y, а кількість входів рекомендується вибирати не більше ніж п'ять. Таблиця істинності повинна описувати всі можливі стани роботи ЛС, тому кількість рядків такої таблиці дорівнює кількості можливих станів. Якщо ЛС є частково визначеною, то значення вихідної функції позначається, як ~.

На другому етапі відбувається подання описування роботи ЛС в аналітичному виді – у вигляді досконалої диз'юнктивної нормальної форми (ДДНФ), або у вигляді досконалої кон'юнктивної нормальної форми (ДКНФ). Вибір форми подання можливо обумовити співвідношенням кількості нулів та одиниць у вихідному сигналі (логічній функції) і їх розташуванням за номерами рядків таблиці істинності або технічним завданням на проведення синтезу схеми.

На третьому етапі відбувається мінімізація схеми ЦП. Найбільш просто і наочно рекомендується проводити мінімізацію за допомогою координатних діаграм (карти Карно або діаграми Вейча). Різниця між цими способами полягає лише у поданні системи координат на діаграмі. На рис. 3.4,а показано вид карти Карно, а на рис. 3.4,б – вид діаграми Вейча.

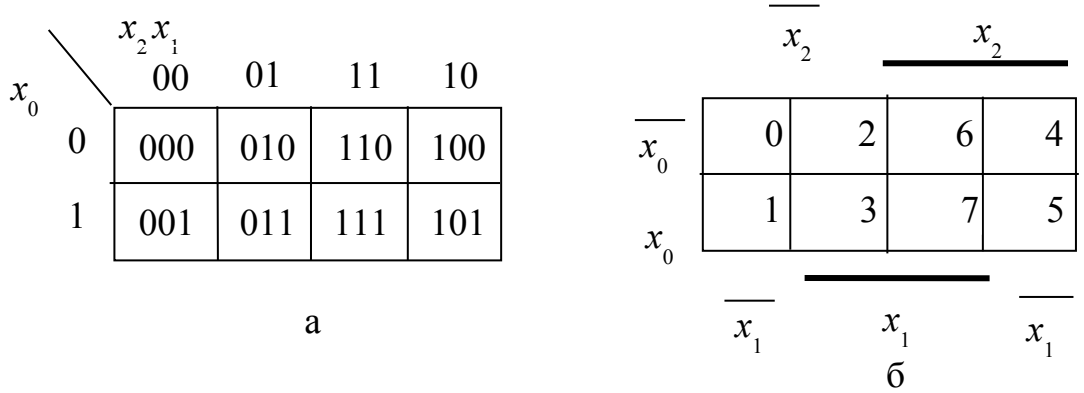


Рисунок 3.4 – Координатні діаграми

Кількість можливих комбінацій значень вхідних сигналів (логічних змінних) *X,* а також кількість можливих стійких станів (логічних функцій) дорівнює $K=2^N$, де N – кількість вхідних сигналів. Слід зазначити, що деякі стійкі стани можуть бути відсутніми, тобто вихідний сигнал Y (логічна функція) індиферентний до комбінації вхідних сигналів X_i у цей час. Це можливо трактувати, як відсутність деяких комбінацій вхідних сигналів X_i у повному наборі можливих. Так, наприклад, дешифратор для керування цифровим індикатором, який відображує десять десяткових цифр 0...9, має чотири входи для подання кодів двійкових чисел від 0000 до 1001, а кількість стійких станів для такої схеми дорівнює $K = 2^4 = 16$. Таким чином, шість комбінацій сигналів ($16_D - 10_D = 6_D$) не можуть бути сформовані на виходах такого пристрою. Значення сигналу карти Карно (рис. 3.4,а) і діаграми Вейча (рис. 3.4,б) для трьох вхідних змінних x_0 , x_1 , x_2, при однаковому розміщенні вхідних змінних на сторонах прямокутників, які є осями координат (загалом, вибір системи координат може бути довільним, обов'язковим є тільки зсув координат на одну клітинку відповідно одна до одної на тих напрямках, на яких необхідно розміщувати дві змінні). У клітинках на рис. 3.4,а показано набори вхідних сигналів, а на рис. 3.4,б номери рядків таблиці істинності, які відповідають один одному.

У результаті мінімізації отримуємо аналітичну функцію, яка складається з мінімальної кількості логічних функцій, за допомогою яких вона реалізується у вигляді мінімальної диз'юнктивної нормальної форми (МДНФ) або мінімальної кон'юнктивної нормальної форми (МКНФ).

Критеріями оптимальності для цих способів є розробка схеми ЛС з мінімальною кількістю логічних елементів і мінімальною кількістю з'єднань між ними, внаслідок чого схема буде споживати мінімум енергії і вартість її буде менше, ніж у інших можливих.

На четвертому етапі за отриманими МДНФ або МКНФ можливо побудувати принципову схему ЦП у базисі ТА-АБО-НІ. Якщо принципову схему необхідно мати в іншому базисі, то виконуємо необхідне перетворення і будуємо схему у відповідному базисі ТА-НІ (АБО-НІ). За необхідності на цьому етапі можливо зробити розрахунок часу затримки сигналу й інших параметрів схеми.

Як приклад зробимо синтез схеми дешифратора для керування одним з елементів семисегментного індикатора, який відображує десяткові цифри 0....9.

1 Таблиця істинності, що завдає алгоритм роботи цього дешифратора, – табл. 3.4.

Таблиця 3.4 – Таблиця істинності дешифратора

№ набору вхідних змінних	x_3	x_2	x_1	x_0	Y
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1

Закінчення табл. 3.4

№ набору вхідних змінних	x_3	x_2	x_1	x_0	Y
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	~
11	1	0	1	1	~
12	1	1	0	0	~
13	1	1	0	1	~
14	1	1	1	0	~
15	1	1	1	1	~

2 За даними цієї таблиці запишемо вираз у вигляді досконалої диз'юнктивної нормальної форми (ДДНФ) або у вигляді досконалої кон'юнктивної нормальної форми (ДКНФ). Досконалість логічних функцій передбачає включення у вираз, який отримується, усіх логічних функцій, які використовуються для формування логічної функції. Будь-який з цих двох способів однозначно описує алгоритм роботи ЛС.

Для подання у вигляді ДДНФ вибираємо рядки таблиці, на яких значення вихідної функції Y дорівнює $L1$ (логічній одиниці) і записуємо у вигляді кон'юнкції (логічного добутку) набори вхідних змінних кожного рядка, які об'єднуємо у вигляді диз'юнкції (логічної суми). Математичний вираз, що описує функцію на одному рядку, називається **термом**. Вхідні змінні повинні входити у кон'юнкцію таким чином, щоб значення логічного добутку дорівнювало $L1$, для чого вхідні змінні, які мають значення $L0$ (логічного нуля) необхідно інвертувати. Так, для нульового набору (рядок таблиці з номером 0) кон'юнкція вхідних змінних матиме вигляд:

$$Y_0 = \bar{x}_3 \wedge \bar{x}_2 \wedge \bar{x}_1 \wedge \bar{x}_0$$

Рядки таблиці, на яких логічна функція не визначена, не беруться до уваги. Тоді логічна функція у вигляді ДДНФ буде записана як

$$Y_{\text{ДДНФ}} = \left(\overline{x_3} \wedge \overline{x_2} \wedge \overline{x_1} \wedge \overline{x_0}\right) \vee \left(\overline{x_3} \wedge \overline{x_2} \wedge x_1 \wedge \overline{x_0}\right) \vee \left(\overline{x_3} \wedge x_2 \wedge x_1 \wedge \overline{x_0}\right) \vee \left(x_3 \wedge \overline{x_2} \wedge \overline{x_1} \wedge x_0\right).$$

Для подавання логічної функції у вигляді ДКНФ, вибираємо рядки, на яких значення вихідної функції Y дорівнює логічному нулю $L0$, і записуємо логічну функцію у вигляді кон'юнкції (логічного добутку) диз'юнкцій вхідних логічних змінних кожного рядка. Для опису вихідної логічної функції Y у вигляді $L1$ логічні зміні кожного рядка, які входять у вираз ДКНФ, необхідно проінвертувати. ДКНФ логічної функції має вигляд:

$$Y_{\text{ДКНФ}} = \left(x_3 \vee x_2 \vee x_1 \vee \overline{x_0}\right) \wedge \left(x_3 \vee x_2 \vee \overline{x_1} \vee \overline{x_0}\right) \wedge \left(x_3 \vee \overline{x_2} \vee x_1 \vee x_0\right) \wedge$$
$$\wedge \left(x_3 \vee \overline{x_2} \vee \overline{x_1} \vee \overline{x_0}\right) \wedge \left(\overline{x_3} \vee x_2 \vee x_1 \vee \overline{x_0}\right).$$

За виразами ДДНФ і ДКНФ можливо оцінити їхню **складність** у вигляді коефіцієнта зв'язку $К_{зв}$, який визначає кількість елементарних логічних функцій, які необхідно виконати для отримання результатів:

$$К_{зв\ ДДНФ} = 4_{НІ} + 4_{ТА} + 1_{АБО} = 9.$$

$$К_{зв\ ДКНФ} = 4_{НІ} + 6_{АБО} + 1_{ТА} = 11.$$

Таким чином видно, що для реалізації функції у ДДНФ необхідно 9, а для реалізації у ДКНФ – 11 логічних елементів.

3 Мінімізацію досконалих логічних функцій виконуємо за допомогою діаграми Вейча, яка для логічної функції з чотирма вхідними змінними матиме вигляд квадрата 4 × 4 клітинки. Діаграма Вейча для логічної функції, яка описана табл. 3.4, наведена на рис. 3.5.

Для мінімізації логічної функції за одиницями або за нулями на діаграмі Вейча необхідно визначити мінімальну кількість найбільших груп (контурів), що складаються з одиниць або нулів, у вигляді квадратів або прямокутників (об'єднувати можливо лише сусідні клітинки). Сусідніми вважаємо клітинки, що мають спільні сторони, при цьому клітинки, розташовані по горизонтальних та вертикальних сторонах квадрата, є сусідніми. Взагалі, ознакою сусідства можливо вважати відміну двійкового коду номера клітинки лише в одному розряді. Таким чином, аналізуючи номери клітинок у верхньому і нижньому рядках, а також у правому і лівому, можливо зробити висновок, що відповідні клітинки у цих рядках також є сусідніми і діаграму Вейча можна згорнути у циліндр по горизонталі і вертикалі. Кількість клітинок у групі повинна дорівнювати цілому ступеню двійки, тобто 1, 2, 4, 8, 16. На рис. 3.5 у вигляді кругів і овалів показано контури, які об'єднують одиниці і нулі поданої логічної функції. Контури об'єднань можуть будь-яку кількість разів перетинатися, але не можуть повністю суміщатися. Якщо функція має невизначені стани (символ – ~), то функції цієї клітинки можливо надати значення 0 або 1 таким чином, щоб отримати контур, до якого буде входити більша кількість клітинок з однаковими значеннями 0 або 1.

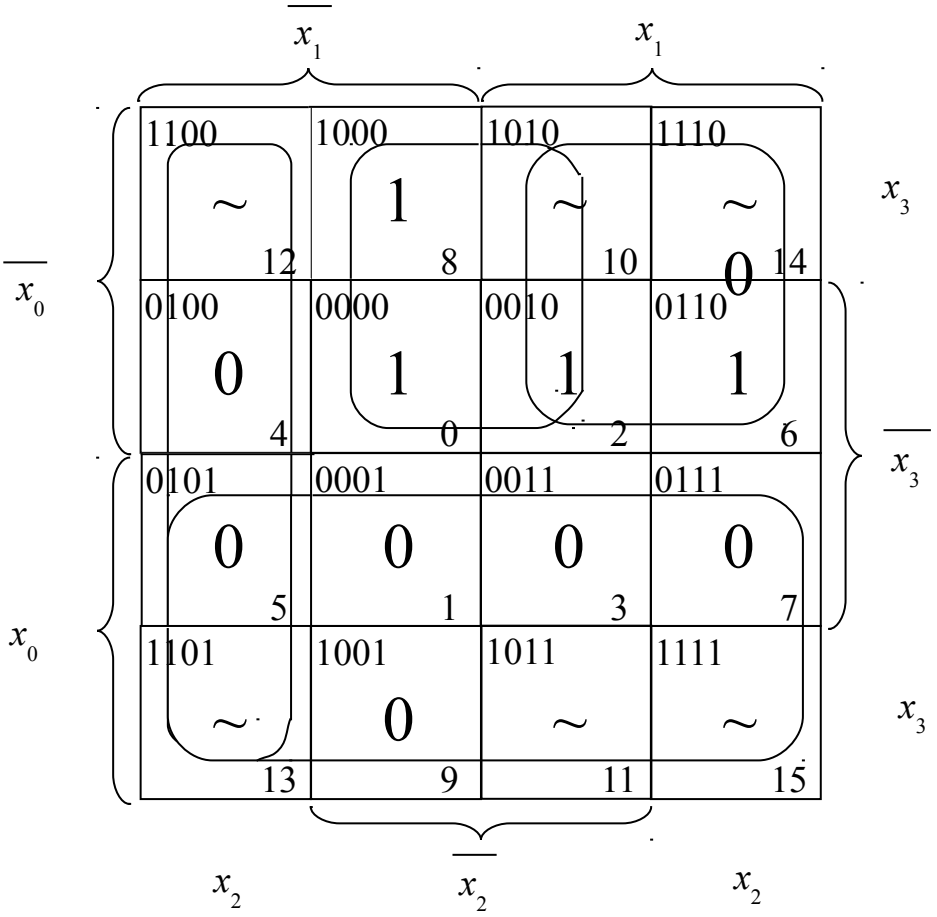


Рисунок 3.5 – Діаграма Вейча з контурами для одиниць і нулів

Таким чином, об’єднавши 1 і невизначені стани, отримали два контури, до яких входять клітинки з номерами 0, 2, 8, 10 і 2, 6, 10, 14. Мінімальною формою логічної функції для кожного з контурів будуть координати контурів на діаграмі з відсутніми координатами, на яких функція Y змінює своє значення. Так, координата x_1 у контурі з номерами клітинок 0, 2, 8, 10 змінює своє значення; у стовпчику з клітинками 0, 8 вона має значення 0, а у стовпчику з номерами клітинок 2, 10 – 1. Таким чином, координата x_1 буде відсутня при записуванні мінімальної ДНФ (МДНФ).

Запишемо мінімальні логічні функції, які отримуємо після мінімізації

$$Y_{\text{МДНФ}} = \left(\overline{x_2} \wedge \overline{x_0}\right) \vee \left(x_1 \wedge \overline{x_0}\right),$$

$$Y_{\text{МКНФ}} = \left(x_2 \vee \overline{x_1}\right) \wedge x_0.$$

Оцінимо $К_{зв}$ для мінімізованих функцій:

для МДНФ $К_{зв}$ = 2НІ + 2ТА + 1АБО = 5,

для МКНФ $К_{зв}$ = 1НІ + 1АБО + 1ТА = 3.

4 По одному з отриманих у попередньому пункті виразах МДНФ або МКНФ можемо побудувати схему логічного пристрою у базисі ТА-АБО-НІ. Критерієм вибору є менше значення $К_{зв}$ або завдання на розробку пристрою.

Логічні схеми будують, як правило, на елементах одного типу в базисах ТА-НІ, АБО-НІ. Це робиться для того, щоб запобігти розкиду параметрів часових затримок у схемі.

Для перетворення виразу до необхідного базису використовується правило подвійної інверсії $\overline{\overline{x}} = x$ та закон де Моргана у такому вигляді:

$$\overline{x_1 \vee x_0} = \overline{x_1} \wedge \overline{x_0}$$
$$\overline{x_1 \wedge x_0} = \overline{x_1} \vee \overline{x_0}$$

Простіше перетворювати вираз МДНФ до базису І-НІ, а вираз МКНФ до базису АБО-НІ, взагалі можливі будь-які перетворення, але при перетворенні МДНФ до базису АБО-НІ і навпаки МКНФ до базису І-НІ, схеми будуть більші на один логічний елемент.

Виконаємо перетворення МДНФ у базис І-НІ

$$Y_{\text{МДНФ}} = \left(\overline{x_2} \wedge \overline{x_0}\right) \vee \left(x_1 \wedge \overline{x_0}\right) = \overline{\overline{\left(\overline{x_2} \wedge \overline{x_1}\right) \vee \left(x_1 \wedge \overline{x_0}\right)}} = \overline{\overline{\left(\overline{x_2} \vee \overline{x_0}\right)} \vee \overline{\left(x_1 \vee \overline{x_0}\right)}},$$

і перетворення МКНФ у базис АБО-НІ

$$Y_{\text{МКНФ}} = \left(x_2 \vee \overline{x_1}\right) \wedge x_0 = \overline{\overline{\left(x_2 \vee \overline{x_1}\right) \wedge x_0}} = \overline{\overline{\left(x_2 \vee \overline{x_1}\right)} \vee \overline{x_0}}.$$

За цими виразами будується схема дешифратора. На рис. 3.6 схема подана у базисі ТА-НІ, а на рис. 3.7 – у базисі АБО-НІ.

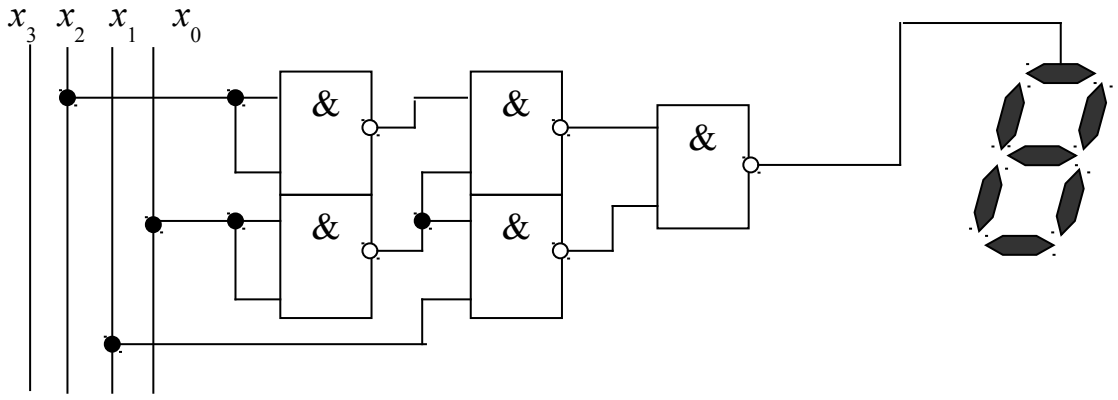


Рисунок 3.6 – Схема дешифратора у базисі ТА-НІ

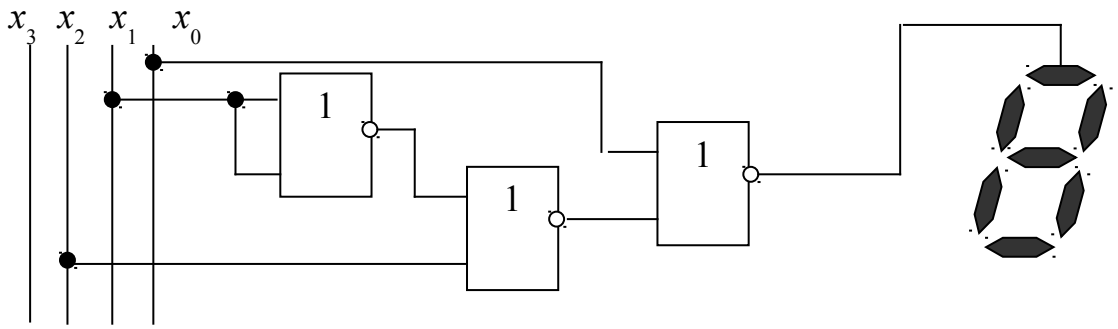


Рисунок 3.7 – Схема дешифратора у базисі АБО-НІ

Час затримки сигналів розраховується за найдовшим шляхом (за максимальній кількості мікросхем, через які проходить хоча б один з сигналів) і становить

$$T_{\text{затр}} = t_{\text{з.м}} \times N_{\text{мс}},$$

де $T_{\text{затр}}$ – час затримки сигналів схемою;

$t_{\text{з.м}}$ – час затримки сигналів однією мікросхемою (визначається технологією виготовлення мікросхеми);

$N_{\text{мс}}$ – максимальна кількість мікросхем від входу схеми до її виходу.

Для розроблених схем

$$T_{\text{затр}} = t_{\text{з.м}} \times N_{\text{мс}} = 20 \cdot 10^{-6} \times 3 = 60 \cdot 10^{-6} \text{ с},$$

де $t_{\text{з.м}} = 20 \cdot 10^{-6}$ секунди – часова затримка однієї мікросхеми, вважаючи, що мікросхема виготовлена за технологією ТТЛ;

$N_{\text{мс}} = 3$ – кількість мікросхем, через які проходять сигнали x_2 і x_0 на рис 3.6, а також сигнал x_1 на рис. 3.7.

Контрольні запитання:

1 Синтезувати схему цифрового пристрою алгоритм роботи якого задано не повністю визначеною логічною функцією, що подана у вигляді таблиці істинності:

№ набору	X_3	X_2	X_1	X_0	H
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1

11	1	0	1	1	1
12	1	1	0	0	0
14	1	1	1	0	0
15	1	1	1	1	1

Визначити складність логічної функції.

Контрольні запитання підвищеної складності:

1 Виконати мінімізацію логічної функції, що подана у вигляді:

$$Y = (x_3 \wedge x_2 \wedge x_1) \vee (\overline{x_3} \wedge x_2 \wedge x_1) \vee (x_3 \overline{\wedge} x_2 \overline{\wedge} x_1) \overline{\vee} (\overline{x_3} \wedge x_2 \wedge x_1)$$

за допомогою координатної діаграми і результат подати у вигляді схеми, що реалізована:

– у базисі ТА–НІ;

– у базисі АБО–НІ.

2 Визначити складність логічної функції до і після мінімізації для кожного з базисів.

3.3 Розробка ЦА

Вхідний контроль:

1 Які цифрові пристрої називаються комбінаційними?

2 Відмінності комбінаційних пристроїв від послідовністних?

3 Яка елементна база використовується для побудови комбінаційних і послідовнісних пристроїв?

Необхідно розробити ініціальний синхронний ЦА (пристрій керування), на основі жорсткої логіки, призначений послідовно формувати чотири дев'ятирозрядні вихідні сигнали (y_0 – y_8). Вхідними сигналами ЦА є сигнали, які формуються на виходах дешифратора станів (елементи *DC*1 – *DC*4), при обробці сигналів стійких станів Q_0 – Q_3 (відповідно z_0 – z_3, у табл. 3.1, 3.2). Таким чином, схема КС1 на рис. 3.1 є схемою дешифратора станів і вхідною логікою керування запам'ятовувальними комірками.

Як запам'ятовувальний пристрій КС1 використовується чотирирозрядний паралельний регістр, який побудовано на *JK*-тригерах. Використання таких тригерів обумовлено наявністю в них досить розвиненої схеми формування вхідних сигналів, що спрощує побудування логічної схеми КС1.

Після формування чотирьох вихідних сигналів ЦА повинен повертатися у нульовий стан. При цьому слід враховувати, що формування першого стану (першої мікрокоманди пристрою керування) і скидання ЦА здійснюється асинхронно.

Для формування тактових імпульсів у схемі необхідно передбачити схему формування цих імпульсів і схему керування їх надходженням на входи синхронізації запам’ятовувального пристрою.

Алгоритм роботи цього ЦА наведено у табл. 3.1 – 3.3 і на рис. 3.2 – 3.3.

Приклад побудови такого ЦА подано рис. 3.8.

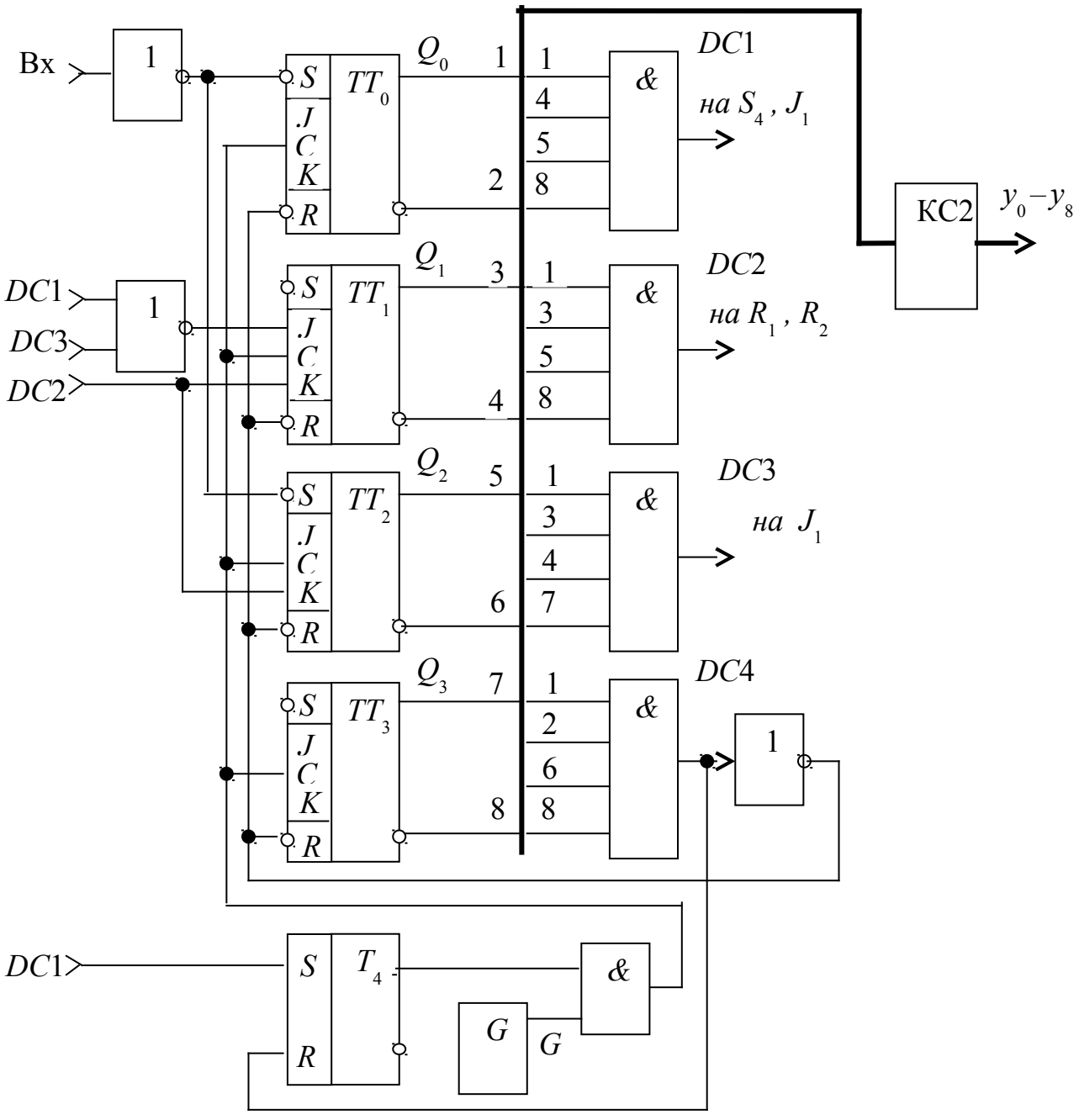


Рисунок 3.8 – Схема цифрового автомата

Розробка принципової схеми такого ЦА передбачає правильне з’єднання елементів для подання сигналів станів по ланцюгах зворотного зв’язку і синтез схеми КС2.

Таким чином, таблиця переходів (табл. 3.1) може бути використана для розробки ланцюгів зворотного зв’язку у вигляді табл. 3.5.

Таблиця 3.5 – Таблиця переходів ЦА

Стани ЦА, Q	Побітне значення алфавіту станів ЦА				Перехід	Адреса подавання сигналів зворотного зв'язку
	Q_3	Q_2	Q_1	Q_0		
Нульовий	0	0	0	0	$0 \rightarrow 5$	Вх $\rightarrow S_0, S_2$
5	0	1	0	1	$5 \rightarrow 7$	$DC1 \rightarrow J_1, S_4$
7	0	1	1	1	$7 \rightarrow 9$	$DC2 \rightarrow K_1, K_2$
9	1	0	0	1	$9 \rightarrow 11$	$DC3 \rightarrow J_1$
11	1	0	1	1	$11 \rightarrow 0$	$DC4 \rightarrow R_0, R_1, R_2, R_3, R_4$

Схема, яка побудована відповідно до цієї таблиці зображена на рис. 3.8. На вхід ЦА у деякий момент часу надходить сигнал у вигляді логічної 1, який необхідно записати у регістр як сигнал першого стану. Це легко зробити, подавши цей сигнал на асинхронні входи тригерів TT_0 і TT_2. Тому що, як активним рівнем сигналу для асинхронних входів є рівень логічного 0, то вхідний сигнал необхідно проінвертувати, для чого використовується інвертор. Запис цього сигналу переведе регістр у стійкий стан – 5, після дешифрування якого буде сформовано сигнал на виході дешифратора $DC1$, який надійде на вхід S_4 тригера T_4 і переведе його в одиничний стан, що дозволить імпульсам тактової частоти надходити на синхровходи C регістра станів. Крім того, цей сигнал надійде на вхід J_1 і підготує його до переключення в одиничний стан при надходженні тактового імпульсу. Сигнали стану по шині надходять на входи КС2, яка сформує на своїх виходах сигнали відповідно до табл. 3.2. Слід зазначити, що на вхід J_1 сигнал керування повинен надходити двічі (згідно з табл. 3.2, стани 5 і 9), тому для організації підведення двох сигналів використовується логічний елемент АБО.

Зміна стану ЦА відбувається при надходженні на синхровходи C негативного перепаду імпульсу тактової синхронізації, відповідно сигналів, які діють на синхронних керуючих входах J і K тригерів регістра стану.

При дешифруванні сигналу 4 стану на виході $DC4$ формується сигнал, який надходить на вхід R тригера T_4 і переводить його у нульовий стан. Це забороняє надходження сигналів тактової синхронізації на входи регістра станів. Сигнал з виходу $DC4$ також надходить на вхід інвертора, з виходу якого – на входи асинхронного скидання усіх тригерів регістра станів. При цьому, ЦА встановлюється в нульовий стан, який буде утримуватись до надходження наступного вхідного імпульсу. Тривалість ЦА в 11 стані визначається швидкістю формування сигналу скидання і надходження його на відповідні входи R_0, R_1, R_2, R_3.

Часові діаграми, що описують роботу ЦА для одного робочого циклу наведено на рис. 3.9.

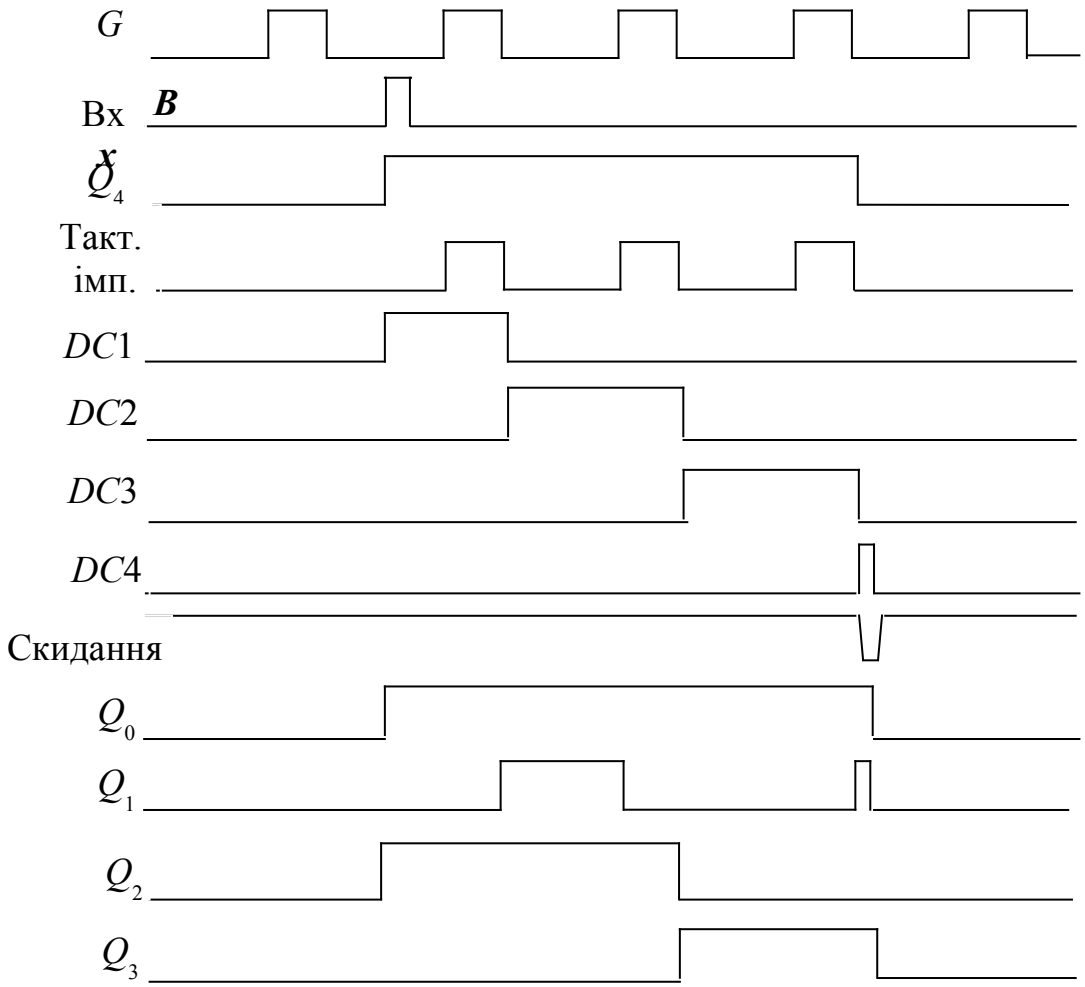


Рисунок 3.9 – Часові діаграми для опису роботи ЦА

Табл. 3.2 визначає значення всіх сигналів $y_0 - y_8$ для кожного зі станів у табличному вигляді. Виконаємо синтез логічної схеми відповідно до цієї таблиці. При аналізі значення окремих стовпчиків $y_0 - y_8$ видно, що значення стовпчика y_1 дорівнює значенню Q_0, значення стовпчиків y_2, y_4, y_7 співпадають, також співпадають значення стовпчиків y_0 і y_6. Таким чином, необхідно зробити синтез 5 різних логічних схем.

Усі функції є частково визначеними, тому щодо інших комбінацій, які не наведено у таблиці, значення функцій y є індиферентними.

Нанесемо функцію, яка відповідає y_2, y_4, y_7 на діаграму Вейча (рис 3.10,а) і виконаємо її мінімізацію. Розміщення координат на діаграмі Вейча вибираємо таким, як на рис. 3.5. З урахуванням невизначених станів можливо об’єднати сусідні групи клітинок, які складають верхній і нижній рядки діаграми. Тоді значення логічної функцій y_2, y_4, y_7 дорівнює Q_3

$$y_2, = y_4 = y_7 = Q_3.$$

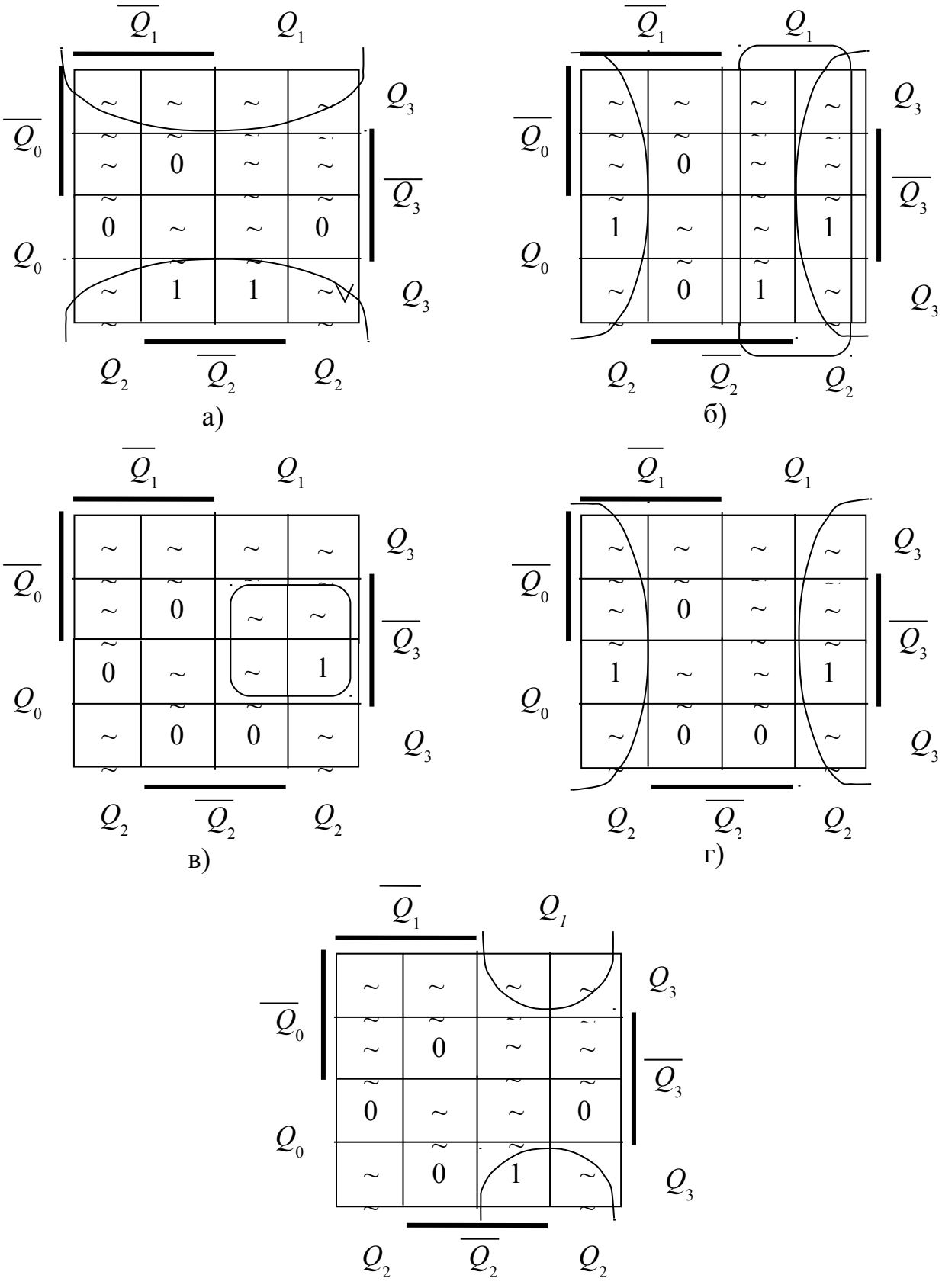


Рисунок 3.10 – Діаграми Вейча для мінімізації логічних функцій, які описують роботу пристрою КС2

На рис. 3.10,б показана діаграма Вейча для мінімізації логічної функції y_0, y_6 . Значення цієї функції у МДНФ становить

$$y_0 = y_6 = Q_2 \vee Q_1 .$$

Аналогічно виконуємо мінімізацію інших функцій. На рис 3.10,в показана діаграма Вейча для логічної функції y_3, на рис. 3.10,г – для функції y_5 і на рис. 3.10,д – для функції y_8 . Ці функції у МДНФ матимуть значення:

$$y_3 = \overline{Q_3} \wedge Q_1 ,$$

$$y_5 = Q_2 ,$$

$$y_8 = Q_3 \wedge Q_1 ,$$

За отриманими значеннями будуємо схему пристрою КС2. Схема зображена на рис. 3.11. Нумерація проводів у шині зберігається такою як і на рис. 3.8.

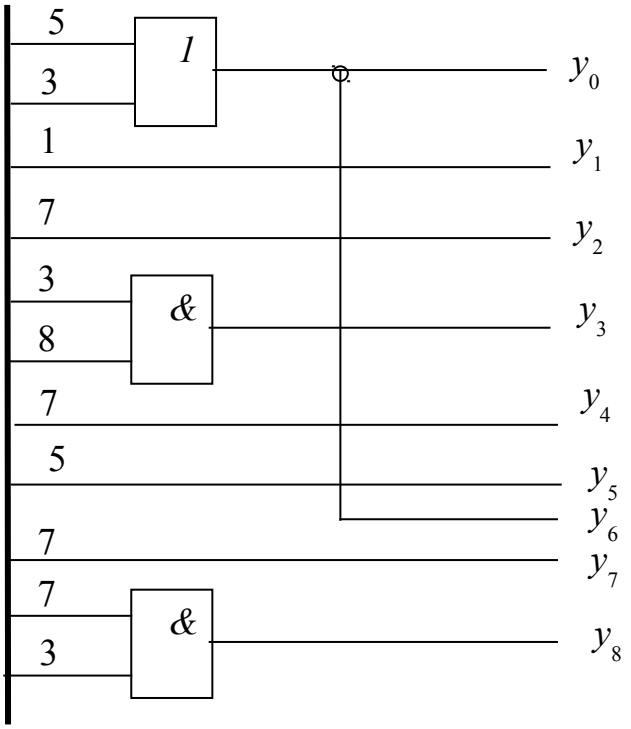


Рисунок 3.11 – Схема пристрою КС2 цифрового автомата

Контрольні запитання:

1 Побудувати схему з’єднання чотирирозрядного регістра станів (ЗП) з входами дешифратора станів ініціального частково визначеного ЦА, який працює за алгоритмом, який наведено у таблиці переходів.

Стани ЦА,	Побітне значення	Перехід

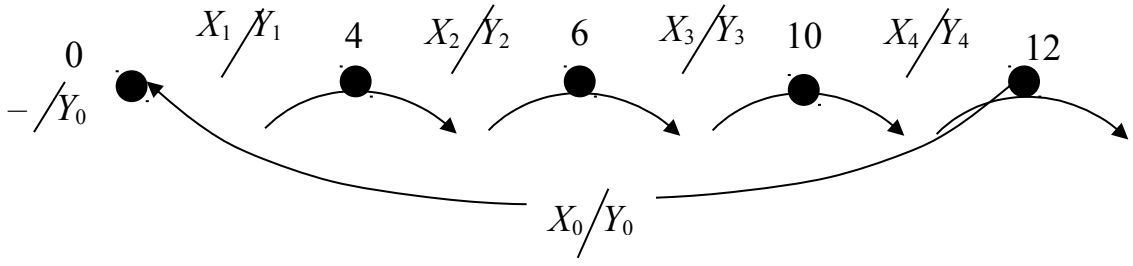
Q	алфавіту станів ЦА				
	Q_3	Q_2	Q_1	Q_0	
Нульовий	0	0	0	0	$0 \rightarrow 3$
3	0	0	1	1	$3 \rightarrow 8$
8	1	0	0	0	$8 \rightarrow 5$
5	0	1	0	1	$5 \rightarrow 12$
12	1	1	0	0	$12 \rightarrow 0$

2 Побудувати схему пристрою КС2 відповідно до наступної таблиці виходів

Стани ЦА, Z	Побітне значення алфавіту станів ЦА				Вихід-ний алфавіт	Побітне значення алфавіту вихідних сигналів								
	Q_3	Q_2	Q_1	Q_0		y_8	y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0
Z_0	0	0	0	0	Y_0	0	0	0	0	0	0	0	0	0
Z_1	0	0	1	1	Y_1	0	1	1	1	0	0	0	1	1
Z_2	1	0	0	0	Y_2	0	0	1	1	0	1	0	0	1
Z_3	0	1	0	1	Y_3	1	1	0	0	1	0	1	1	0
Z_4	1	1	0	0	Y_4	1	1	1	0	0	0	1	0	1

Контрольні запитання підвищеної складності:

1 Розробити таблицю виходів цифрового автомата, алгоритм роботи якого задано у вигляді спрямованого графа



Значення сигналів вхідного і вихідного алфавітів вибрати самостійно у вигляді чотирирозрядного двійкового числа для вхідного алфавіту і п’ятирозрядного двійкового числа – для вихідного.

4 ТИПОВІ ПРИСТРОЇ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ
(Для самостійного вивчення)

Вхідний контроль:

1 Які типові цифрові пристрої Ви знаєте?
2 Чим відрізняються комбінаційні і послідовнісні пристрої?
3 Які елементи є основою для побудови цифрових пристроїв?

До типових пристроїв мікропроцесорних систем можна віднести такі пристрої: двійкові суматори, цифрові компаратори, арифметично-логічний пристрій, запам'ятовувальні пристрої, пристрої введення-виведення інформації, таймери, програмовні логічні інтегральні схеми (ПЛІС) тощо.

4.1 Суматори

Суматор – це вузол обчислювальної системи, що виконує арифметичне додавання кодів двійкових чисел. Суматор будується з комбінаційних схем, які виконують додавання двох однорозрядних двійкових чисел *a* і *b*, формують однорозрядний сигнал їхньої суми – *S* і сигнал перенесення в наступний старший розряд – *C*. Такі пристрої називають напівсуматорами і вони є елементною базою для побудови повних суматорів. Умовне графічне позначення такої схеми показано на рис. 4.1.

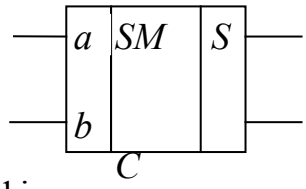


Рисунок 4.1 – Умовне графічне позначення комбінаційного напівсуматора

Схема повного однорозрядного суматора буде відрізнятися тим, що одним із вхідних сигналів, додатково до операндів *a* і *b*, буде значення вхідного перенесення, яке буде додаватися до результату. Алгоритм роботи повного однорозрядного суматора можливо описати за допомогою таблиці істинності – табл. 4.1.

Таблиця 4.1 – Таблиця істинності однорозрядного суматора

Вхідне перенесення *c*	Додаток *a*	Додаток *b*	Вихідне перенесення *C*	Сума *S*
0	0	0	0	0
0	1	0	0	1
0	0	1	0	1
0	1	1	1	0

Закінчення табл. 4.1

Вхідне перенесення c	Додаток a	Додаток b	Вихідне перенесення C	Сума S
1	0	0	0	1
1	1	0	1	0
1	0	1	1	0
1	1	1	1	1

За даними цієї таблиці можливо виконати синтез схеми повного суматора. Після чого буде видно, що ця схема є поєднанням схем двох напівсуматорів. Схему повного суматора показано на рис. 4.2.

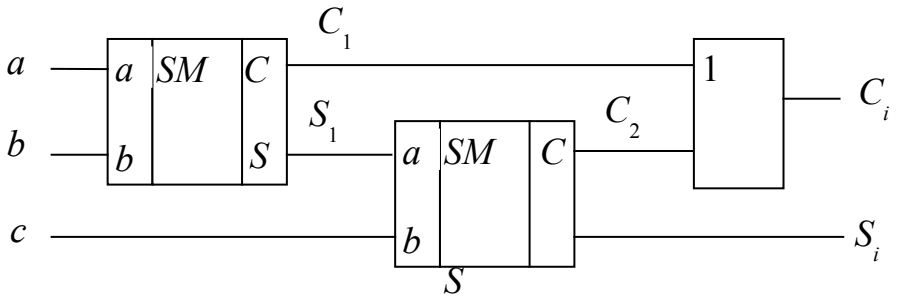


Рисунок 4.2 – Схема повного однорозрядного суматора

У цій схемі формуються два проміжні перенесення і одна проміжна сума, які далі беруть участь у формуванні результату. Вихідне перенесення формується як результат виконання логічної функції АБО сигналів двох проміжних перенесень.

Багаторозрядні суматори будуються як схеми паралельного з’єднання повних однорозрядних суматорів з послідовними лініями міжрозрядних перенесень. Приклад схеми такого суматора показано на рис. 4.3.

Комбінаційний суматор можливо також використовувати для виконання операції віднімання чисел зі знаками, для чого операнди, що надходять на схему, повинні подаватися у доповнювальному коді.

Важливою рисою комбінаційного паралельного суматора є затримка у колах формування перенесень. За характером розповсюдження перенесення суматори поділяють на три класи: суматори з порозрядним паралельним перенесенням, з паралельним перенесенням і з груповим перенесенням.

Паралельні суматори будуються за схемою, що показана на рис. 4.3, і характеризуються послідовним розповсюдженням сигналу перенесення від розряду до розряду, за ступенем формування результату. Затримка формування суми, за наявності перенесень буде дорівнювати

$$T_{\text{пер}} = \tau \cdot n$$

де τ – затримка сигналу в одному розряді суматора; n – кількість розрядів. Видно, що ця величина швидко зростатиме зі збільшенням кількості розрядів.

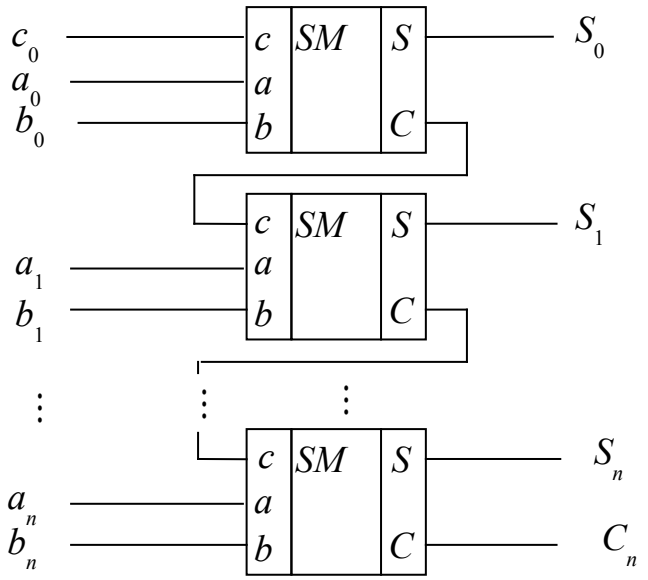


Рисунок 4.3 – Схема *n*-розрядного комбінаційного паралельного суматора

Таким чином, такий тип суматора характеризуються простотою схеми формування перенесення, але має низьку швидкодію.

Для усунення цього недоліку, можливо використовувати схему суматора з паралельним перенесенням. В такому суматорі додавання виконується як порозрядна операція і вхідне перенесення для кожного старшого розряду формується незалежно від формування перенесення у попередньому молодшому розряді. Для всіх розрядів сигнали перенесення також формуються паралельно. Формування сигналів перенесення відбувається в результаті формування й оброблення додаткових двох сигналів: розповсюдження перенесення (*CRP – Carry Propagation*) і генерування перенесення (*Carry Generation*). Затримка отримання суми в такому суматорі складається із однакових значень затримки перенесення для всіх розрядів, які не залежать від кількості розрядів. Слід зазначити, що апаратні затримки в такій схемі швидко зростають відповідно до збільшення кількості розрядів, тому в чистому вигляді такий тип суматора майже не використовується. Для невеликої кількості розрядів (до 8) промисловість випускає спеціальні мікросхеми – схеми прискореного перенесення.

Для прискорення формування сигналів перенесення у суматорах зі значною кількістю розрядів використовується принцип групового перенесення, відповідно до якого розрядна сітка розбивається на декілька груп з однаковою кількістю розрядів у кожній. Кожна група являє собою суматор, усередині якого перенесення може формуватися як послідовно так і паралельно, а перенесення із групи в групу відбувається по тракту міжгрупового перенесення, який може буде побудований як паралельний, так і послідовний. У паралельному тракті перенесення між групами формуються як функції лише доданків. У послідовному тракті сигнал перенесення формується в кожній групі і з виходу молодшої групи надходить на вхід перенесення наступної старшої групи.

Використання паралельного перенесення всередині групи з паралельним перенесенням між групами дозволяє будувати найбільш швидкодіючі суматори з розрядністю 24 – 64 біти. Взагалі, вибір схеми суматора і формування перенесення в кожному випадку виконується за результатами аналізу апаратних витрат і забезпеченням необхідної швидкодії.

Суматори є елементною базою для побудови цифрових компараторів і арифметично-логічних пристроїв.

Контрольні запитання:

1 Яке призначення схеми суматора?
2 Які умови формування сигналів на виходах однорозрядного суматора?
3 В чому полягає причина низької швидкодії паралельного суматора?
4 Якими способами можливо підвищити швидкодію суматора?

Контрольні запитання підвищеної складності:

1 В чому полягає принцип групового перенесення?
2 Поясніть причину формування затримки сигналу у схемі паралельного суматора?

4.2 Цифрові компаратори

Вхідний контроль:

1 Який пристрій називається компаратором?
2 Для чого використовується компаратор?

Цифровим компаратором називають функціональний вузол обчислювальної техніки, який порівнює два багаторозрядних числа A і B між собою і результат порівняння подає у вигляді сигналів співвідношення між ними $A = B$, $A < B$, $A > B$.

Для порівняння сигналів використовується суматор, який виконує операцію віднімання (алгебраїчного додавання) вхідних чисел і за цими результатами формуються сигнали, що показують співвідношення між ними. Для виконання операції віднімання одне із вхідних чисел подається у доповнювальному коді на відповідні входи суматора, тому до складу цифрового компаратора входить схема, яка формує доповнювальний код одного з операндів. За результатами виконання операції $A – B$ можна зробити такі висновки:

– якщо результат дорівнює 0, то це є ознакою того, що значення A і B співпадають;

– якщо результат не дорівнює 0, а вихідне переповнення дорівнює 1, то $A < B$;

– якщо вихідне переповнення дорівнює 0, то це є ознакою, що $A > B$.

Сигнали, що показують співвідношення між вхідними числами формуються за допомогою логічних схем. Функціональна схема цифрового компаратора для чотирирозрядних чисел наведена на рис. 4.4.

Умовне графічне позначення компаратора, який показано на рис. 4.4, наведено на рис. 4.5.

Для нарощування розрядності вхідних даних цифрові компаратори дозволяється каскадувати, для чого набір вхідних сигналів компаратора доповнюють вихідними сигналами попереднього каскаду $A = B$, $A < B$, $A > B$ (рис. 4.5). Формування кінцевого результату порівняння відбувається з урахуванням цих сигналів.

Рисунок 4.4 – Функціональна схема чотирирозрядного цифрового компаратора

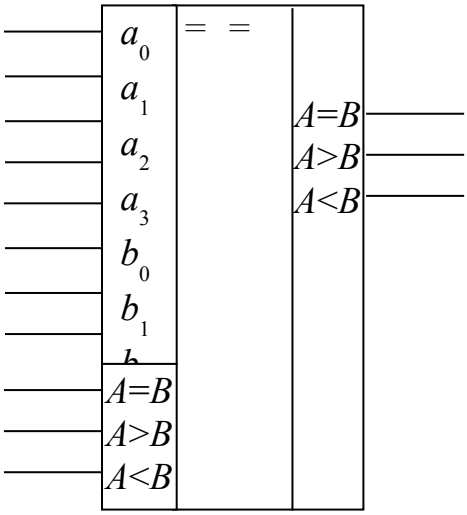


Рисунок 4.5 – Умовне графічне позначення чотирирозрядного цифрового компаратора

Контрольні запитання:

1 Як виконується порівняння двох двійкових чисел у цифровому компараторі?

2 Поясніть призначення формувача доповнювального коду у схемі, що наведена на рис. 4.4.

Контрольні запитання підвищеної складності:

1 На підставі якої ознаки приймається рішення, що $A = B$?

2 На підставі яких ознак приймається рішення, що $A > B$?

4.3 Арифметично-логічний пристрій

Вхідний контроль:

1 Які арифметичні операції використовуються в обчислювальній техниці?

2 Які логічні операції використовуються в обчислювальній техніці?

3 Які правила використовуються при виконанні арифметичних операцій в обчислювальній техниці?

Арифметично-логічні пристрої (АЛП) широко використовуються для побудування арифметичних вузлів, зокрема, АЛП є обов'язковою складовою частиною будь-якого процесора. АЛП використовується для виконання арифметичних і логічних операцій з даними, які до нього надходять і являють собою числа або будь-який інший вид інформації. Кількість розрядів у даних, над якими виконується операція, звичайно співпадає з кількістю розрядів в основних регістрах МПС. АЛП випускають як окремі мікросхеми або вони є невід'ємною складовою мікросхеми центрального процесора.

Операції, які виконуються в АЛП, можна розділити на такі групи:

- арифметичні операції над двійковими числами з фіксованою точкою;
- арифметичні операції над двійковим числами з плавучою точкою;
- операції десяткової арифметики;
- операції індексної арифметики (для модифікації адрес команд);
- логічні операції з операндами, які є логічними змінними;
- спеціальні арифметичні операції;
- операції над алфавітно-цифровими полями.

До арифметичних операцій відносяться додавання, віднімання, множення і ділення двійкових і двійково-десяткових чисел, операції з рядками даних. До групи логічних операцій входять: диз'юнкція (логічне АБО), кон'юнкція (логічне ТА), виключне АБО, інверсія, логічні зсуви, порівняння кодів між собою тощо. До групи спеціальних арифметичних операцій входять операції нормалізації даних, арифметичні зсуви тощо.

Арифметичні операції в АЛП виконуються за допомогою багаторозрядного суматора, а логічні – відповідними логічними схемами. В залежності від характеру використання цих пристроїв АЛП поділяються на **блочні** і **багатофункційні**. В блочних АЛП для виконання різних типів операцій використовуються окремі блоки. Це дає змогу підвищити швидкодію, але збільшує апаратні витрати. В багатофункційних АЛП для виконання операцій різних типів використовуються одні й ті ж пристрої, які комутуються по-різному, в залежності від виконуваної операції. Вибір операції і необхідних блоків здійснюється за допомогою сигналів керування, якими кодується операція, яку необхідно виконати. Крім результату АЛП формує набір спеціальних сигналів – ознаки результату (прапорці). Умовне позначення АЛП на схемах показано на рис. 4.6.

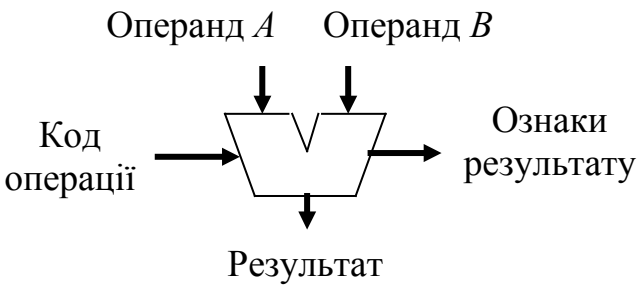


Рисунок 4.6 – Умовне графічне позначення АЛП

Тому що, двійковий суматор, який використовується в АЛП для виконання арифметичних операції, є комбінаційним пристроєм, то для одночасного подання двох операндів на його входи необхідно мати запам'ятовувальні пристрої для тимчасового зберігання операндів і результату.

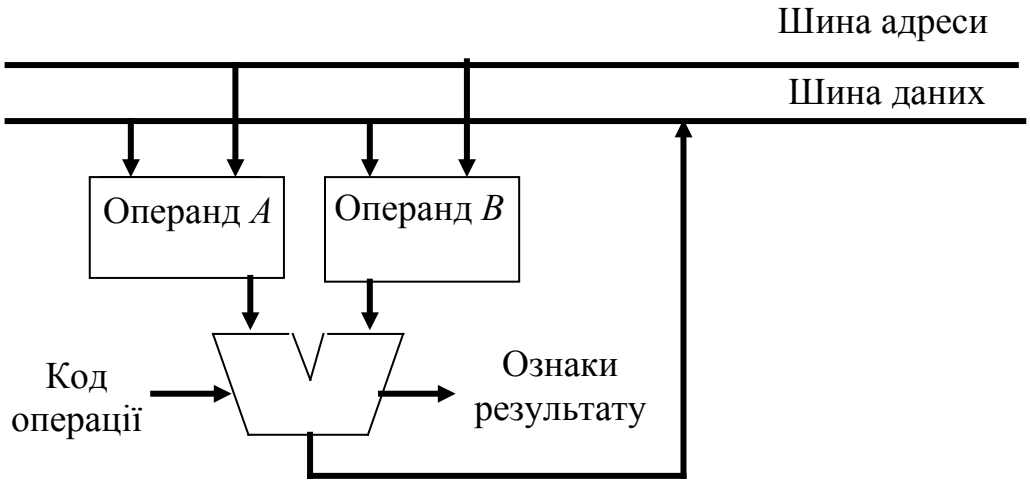
До АЛП операнди можуть надходити з двох регістрів загального призначення (РЗП) або із регістра і комірки пам'яті, а результат надходить до регістру або до комірки пам'яті, які визначені як приймач результату. Така система має назву **двоадресної**. Тому МПС повинна подати адреси обох операндів. Схему організації двоадресної системи показано на рис. 4.7,а.

В інших МПС один з операндів до початку виконання команди обов'язково зберігається в акумуляторі, а після закінчення результат записується на місці операнда, який безповоротно втрачається. Така система має назву **одноадресної**, тому що необхідно адресувати лише один операнд. Схему організації одноадресної системи показано на рис. 4.7,б.

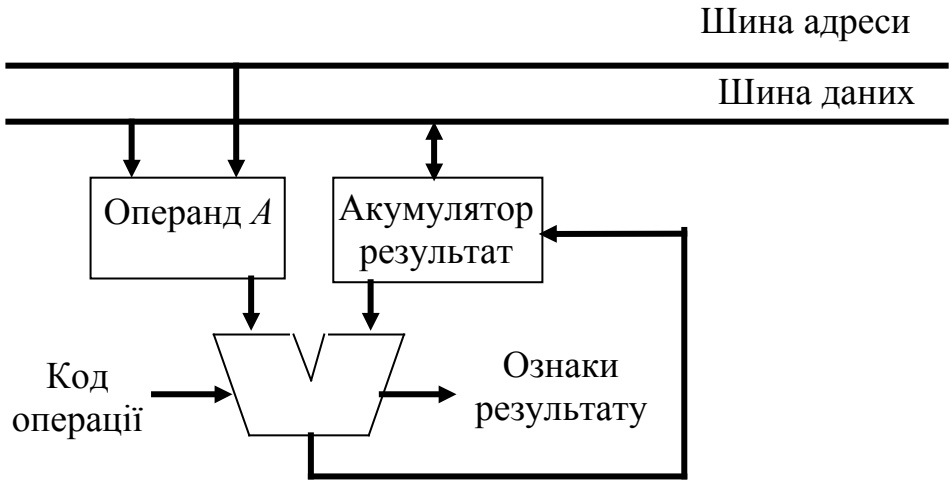
До важливої характерної ознаки АЛП відноситься формування ним ознак результату (прапорців) – тобто деяких властивостей результату, які можуть вплинути на подальше виконання програми. Ознаки результату формуються після отримання результату і записуються у спеціальний регістр – регістр ознак результату (або інші назви цього регістра – регістр прапорців, регістр стану), де вони зберігаються до формування результату наступної операції. Слід зазначити, що ознаки результату формують лише команди арифметичних і логічних операцій. Регістр ознак, у залежності від типу мікропроцесора, може мати різну кількість розрядів, які відповідають властивостям результату і зберігають інформацію про стан деяких апаратних або програмних компонентів процесора. Формат регістра ознак для відповідної мікросхеми може бути таким, як показано на рис. 4.8.

Кожна з ознак результату (прапорець) зберігається у одному розряді регістра і кодує наявність відповідної ознаки значенням 0 або 1. Розглянемо кожну з ознак результату з короткою характеристикою:

– *C* (*carry*) – ознака зберігає значення перенесення до наступного старшого розряду за межами розрядної сітки або значення позики з цього розряду;



а)



б)

Рисунок 4.7 – Схема організації двохадресного (а) і одноадресного (б) АЛП

C	*S*	*Z*	*AC*	*OVR*	*P*
Ознака перенесення (позики)	Ознака знаку	Ознака нуля	Ознака допоміжного перенесення	Ознака переповнення	Ознака додатності

Рисунок 4.8 – Формат регістра і назви ознак результату

– *S* (*sign*) – розряд регістра зберігає копію знакового розряду результату операції. Значення цієї ознаки, що дорівнює 1 відповідає від'ємному результату;

– *Z* (*zero*) – ознака нульового результату. Для результату, що дорівнює 0 – ознака набуває значення 1;

– *AC* (*auxiliary carry*) – ознака зберігає значення перенесення зі старшого розряду молодшої тетради у молодший розряд старшої тетради байта

при додаванні двійково-десяткових чисел або – значення позики зі старшої тетради до молодшої при відніманні;

– *OVR* (*overflow*) – ознака, яка сигналізує про втрату старшого біта результату додавання або віднімання чисел зі знаком, що обумовлено наявністю перенесення у знаковий розряд. При додаванні ознака набуває значення 1, якщо є перенесення у старший (знаковий) біт результату, але немає перенесення з нього і навпаки якщо є перенесення у біт *C,* але немає перенесення у старший (знаковий) біт. При відніманні ознака дорівнює 1, якщо є позика зі старшого (знакового) біта результату, але немає позики з нього і навпаки якщо є позики з біта *C,* але немає позики зі старшого (знакового) біта. Значення ознаки дорівнює результату виконання логічної операції ВИКЛЮЧНЕ АБО над значеннями бітів *C* і перенесенням до старшого (знакового) біта результату

$$C \oplus C_{D7},$$

де C_{D7} – перенесення до старшого (знакового) біта результату.

– *P* (*parity*) – значення цієї ознаки дорівнює 0, якщо результат операції складається з непарної кількості одиниць.

Формування ознак покажемо на прикладі виконання арифметичної і логічної операцій для восьмирозрядних чисел:

– арифметична операція над двійковими числами

$$\begin{array}{r} 11101111 \\ +\ 01111000 \\ \hline 101100111 \\ \overset{\curvearrowleft}{C}\quad \overset{\curvearrowleft}{AC} \end{array}$$

Якщо вважати, що операція виконується над беззнаковими числами, то це $239_D + 120_D = 359_D$ і результат з урахуванням перенесення правильний, якщо вважати що числа зі знаками, то це $(-17_D) + 120_D = +103_D$. Враховуючи, що числа зі знаком подані в доповнювальному коді, то результат також правильний. Ознаки результату не залежать від того над якими числами виконувалась операція. Вони набувають таких значень:

$C = 1$ – перенесення у 9 розряд має місце;

$S = 0$ – восьмий розряд результату, що кодує знак, дорівнює 0, результат додатний;

$Z = 0$ – результат не дорівнює нулю;

$AC = 1$ – відбулося перенесення з молодшої тетради до старшої. Слід сказати, що АЛП виставляє ознаки, формально, за їх наявності, не враховуючи конкретні обставини виконання операції;

$OVR = 1$ – відбулося переповнення розрядної сітки (біт *C*) і не відбулося перенесення до старшого (знакового) розряду.

$$C \oplus D7 = 1 \oplus 0 = 1.$$

Якщо це числа зі знаком і передбачається їх подальша обробка, то необхідно збільшити довжину розрядної сітки;

P = 0 – результат (без урахування перенесення) складається з непарної кількості (5) одиниць.

– логічна операція кон'юнкція над восьмирозрядними логічними змінними

$$\begin{array}{r} 11101111 \\ \wedge\ 01111000 \\ \hline 01101000 \end{array}$$

При виконанні логічних операцій перенесення не відбувається, тому ознаки C, AC і OVR не формуються, а зберігають свої значення, яких набули при попередній операції. Ознака S формується формально копіюванням старшого розряду результату. Після виконання логічної операції ознаки набувають таких значень:

C = X – зберігає попереднє значення;

S = 0 – восьмий розряд результату, що кодує знак, дорівнює 0;

Z = 0 – результат не дорівнює нулю;

AC = X – зберігає попереднє значення;

OVR = X – зберігає попереднє значення;

P = 0 – результат складається з непарної кількості (3) одиниць.

Умовне графічне позначення мікросхеми чотирирозрядного АЛП наведено на рис. 4.9.

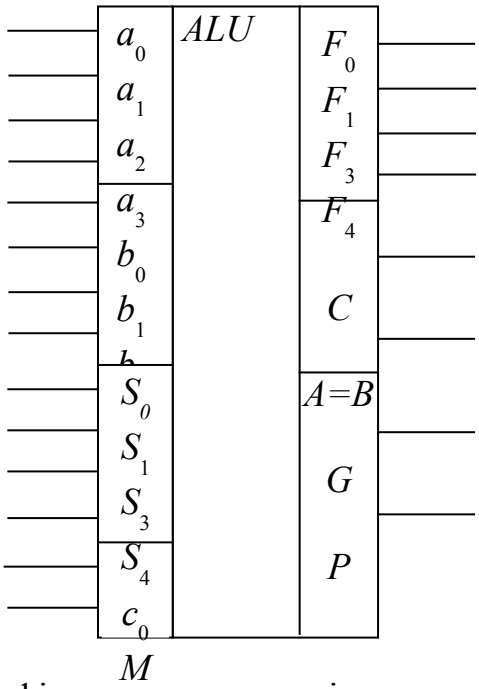


Рисунок 4.9 – Умовне графічне позначення мікросхеми чотирирозрядного АЛП

Виводи цієї мікросхеми мають таке функціональне призначення:

- $a_0 – a_3$ – входи операнда A;
- $b_0 – b_3$ – входи операнда B;
- $S_0 – S_3$ – входи для подання коду операції, що буде виконуватись;

– c – вхідне перенесення;

– M – код режиму роботи M = 1 – АЛП може виконувати 16 логічних операцій, M = 0 – 16 арифметичних операцій;

– $F_0 – F_3$ – результат виконання операції;

– C – вихідне перенесення;

– $A = B$ – ознака рівності вхідних операндів;

– G – вихід сигналу генерування перенесення;

– P – вихід сигналу розповсюдження перенесення.

Останні два сигнали використовуються для нарощування розрядності операндів і збільшення швидкодії схеми, яка буде отримана.

Контрольні запитання:

1 Яке місце займає АЛП у складі мікропроцесора?

2 Чому операнди, над якими необхідно виконати операцію в АЛП, необхідно спочатку записати в два запам'ятовувальні пристрої – регістри?

3 Які ознаки результату формуються після виконання операції в АЛП і яке їх призначення?

4 Які ознаки результату не змінюються при виконанні логічних операцій?

Контрольні запитання підвищеної складності:

1 Які ознаки результату будуть сформовані при виконанні арифметичної операції

$$+89_D - 120_D.$$

2 Які ознаки результату будуть сформовані при виконанні логічної операції

$$11001110 \vee 01100011.$$

3 Яке значення буде мати ознака P, якщо результат виконання операції дорівнює 11100001?

4.4 Програмовані логічні інтегральні схеми (ПЛІС)

У мікропроцесорних системах у багатьох випадках, наприклад, у телекомунікаціях, при виконанні цифрового оброблення сигналів можна доповнювати, а іноді навіть заміняти мікропроцесорні засоби на програмовані логічні інтегральні схеми (ПЛІС).

На них можна будувати спеціалізовані цифрові пристрої, керувальні засоби – контролери, і застосовувати у тих областях, де апаратним рішенням задач можна віддати перевагу порівняно з програмними рішеннями, які завжди є послідовними. Застосування ПЛІС забезпечує паралельне оброблення різних гілок одного обчислювального процесу і, таким чином, збільшує продуктивність системи іноді у десятки разів. У той самий час зберігається така ж сама гнучкість реалізації алгоритмів, як і при програмному способі. Задавати алгоритм дії проектованого цифрового пристрою для реалізації на ПЛІС можна

у вигляді часових діаграм, текстового опису, схем на логічних елементах, у вигляді логічних функцій. Для отримання оптимального алгоритму використовується процедура мінімізації, як було показано вище. Використовуючи засоби САПР, розробники отримують файл, який використовується потім при програмуванні ПЛІС на програматорі. ПЛІС – це надвелика інтегральна схема, яка вміщує на кришталі універсальні налаштовувані користувачем функціональні перетворювачі та програмовані зв'язки між ними. ПЛІС можуть реалізовувати блоки пам'яті, блоки цифрової обробки сигналів, вбудовані процесорні ядра з периферією, швидкісні канали введення-виведення.

Окремі цифрові пристрої – шифратори, перетворювачі кодів, спеціальні функціональні пристрої, декодери адрес тощо, як правило, входять до складу мікропроцесорних систем. Вони не є складними й описати алгоритм їхньої роботи можна у вигляді логічної функції не більше як від трьох до п'яти логічних змінних. Синтез таких логічних схем можна здійснити після мінімізації логічних функцій, які їх описують. Їх можливо реалізовувати на трьох типах програмовних логічних пристроїв:

- програмовані логічні матриці ПЛМ (*PLA – programmable logic array*);
- програмована матрична логіка ПЛМ (*PAL – programmable array logic*);
- вентильна матрична логіка ВМЛ (*GAL – gated array logic*).

Останнім часом елементи *PAL* знайшли широке застосування у МПС для побудування адресних дешифраторів. Схема такого пристрою складається з двох матриць логічних пристроїв – матриці логічних елементів АБО та матриці логічних елементів ТА. Програмування такої структури полягає у перепалюванні перемичок між матрицями, в результаті чого формується схема, що відповідає заданому алгоритму роботи.

Контрольні запитання:

1 Чому використання ПЛІС дозволяє збільшити продуктивність МПС?
2 Які типи мікросхем ПЛІС Ви знаєте?
3 Для чого використовуються мікросхеми *PAL* у МПС*?*

Контрольні запитання підвищеної складності:

1 Поясніть для чого необхідно використовувати мінімізацію при розробці алгоритму робити ПЛІС.

5 ПРИНЦИПИ ПОБУДУВАННЯ ЗАПАМ'ЯТОВУВАЛЬНИХ ПРИСТРОЇВ МПС З ЗАДАНОЮ ОРГАНІЗАЦІЄЮ

Вхідний контроль:

1 Які підсистеми входять до складу МПС?

2 Яке призначення має підсистема пам'яті у складі МПС?

5.1 Запам'ятовувальні пристрої МПС та їх класифікація

Підсистема пам'яті є однією зі складових частин МПС, яка значною мірою визначає її продуктивність і обчислювальні можливості. До підсистеми пам'яті входять технічні пристрої, які називаються **запам'ятовувальними пристроями** (ЗП) і призначені для зберігання двійкової інформації. Основними операціями у пам'яті є запис, зберігання і вибірка (читання) інформації. Сукупність операції запису і вибірки називається **зверненням до пам'яті**.

ЗП будуються з двопозиційних **елементів пам'яті** (ЕП)**,** кожен з яких зберігає один біт інформації. Сукупність декількох елементів пам'яті створюють **комірку пам'яті,** яка призначена для зберігання багаторозрядної двійкової інформації і звернення до елементів якої відбувається одночасно. Звернення до ЗП відбувається за адресним принципом, який передбачає наявність у кожній комірці пам'яті відповідного номера, що називається **адресою** і яку необхідно явно чи неявно вказувати при зверненні. Крім адресних, використовують асоціативні ЗП, звернення до комірок яких відбувається за результатами аналізу деяких розрядів інформації, яка зберігається.

Класифікувати і порівнювати ЗП можливо за багатьма різними критеріями в залежності від потреби користувача і технічних вимог до побудування пам'яті. Основними критеріями, які визначають побудування і функціонування ЗП, є: фізичний принцип роботи запам'ятовувальних елементів і технологія їхнього виготовлення, доступ до комірок пам'яті, швидкість обміну інформацією, спосіб зберігання інформації тощо.

Так, в залежності від природи фізичного середовища, в якому зберігається інформація, ЗП поділяються на напівпровідникові, пристрої із зарядним зв'язком (ПЗЗ), магнітні, оптичні тощо. Надалі будемо розглядати напівпровідникові ЗП.

Для порівняння запам'ятовувальних пристроїв між собою розроблена система параметрів, що характеризує їхні властивості. До таких параметрів відносяться:

– **інформаційна ємність** – це кількість одиниць інформації, що одночасно може зберігатися у ЗП і визначається у двійкових одиницях бітах або байтах. У разі використання одиниці вимірювання – байт, ємність пам'яті подають через число $K = 1024$ байт = 1 кбайт;

– **розрядність даних** – визначається вмістом (кількістю розрядів) комірки пам'яті;

– **організація пам’яті** – це параметр, який поєднує два попередніх у вигляді виразу

$$N \times n,$$

де N – кількість комірок пам’яті, які входять у ЗП; n – розрядність даних.

Добуток, який буде отримано при виконанні множення, буде дорівнювати інформаційній ємності в тій одиниці вимірювання, яка визначена розрядністю даних;

– **час вибірки** – інтервал часу з моменту надходження запиту на передачу даних до моменту їхньої появи на виході ЗП;

– **тривалість циклу звернення (цикл пам’яті)** – мінімальний інтервал часу між двома сусідніми зверненнями до ЗП, кожен з яких буде нормально виконуватись;

– **напруга живлення ЗП** – визначає вибір необхідного пристрою живлення;

– **потужність енергоспоживання** – електрична потужність, яку споживає ЗП при роботі. Для деяких типів ЗП розрізняється потужністю енергоспоживання у режимі звернення (запис, читання) й у режимі зберігання. Потужність енергоспоживання у режимі звернення набагато більша за потужність у режимі зберігання;

– **питома вартість –** визначає співвідношення вартості та інформаційної ємності ЗП.

Вимоги до обсягу і швидкодії пам’яті є суперечними. Досягнення високої швидкодії є досить важкою технічною задачею, вирішення якої потребує додаткових витрат для досягнення необхідного обсягу пам’яті. Взагалі вартість ЗП складає значну частину витрат на апаратну частину МПС. Для раціональної організації пам’яті, відповідно до принципів фон Неймана, передбачено класифікацію ЗП зі швидкодії, яка окреслює певну ієрархію пристроїв пам’яті. Це необхідно для забезпечення високої швидкості обміну інформацією між пам’яттю й арифметично-логічним пристроєм (АЛП) мікропроцесора за великої ємності ЗП МПС. Ієрархія ЗП МПС показана на рис. 5.1. Вершину цієї ієрархії складають найбільшвидкодіючі ЗП певної МПС, які входять до складу центрального процесора – надоперативний ЗП (НОЗП) і внутрішня кеш-пам’ять.

Надоперативний ЗП (НОЗП) складається з регістрів загального призначення ЦП, кількість яких залежить від архітектури процесора і складає від 16 до 64 регістрів, кількість розрядів у кожному з котрих визначається регістровою моделлю процесора. Ці регістри не потребують виставлення адреси та її оброблення при зверненні до них, тому швидкість роботи з ними максимальна.

Внутрішня кеш-пам’ять – це різновид програмно недоступної оперативної пам’яті ємністю 1 – 16 кбайт, яка вбудована у ЦП. В залежності від типу процесора може бути одна кеш-пам’ять, спільна для даних і команд, або дві окремих.

Зовнішня кеш-пам’ять – це також оперативна пам’ять, яка установлюється на системній платі і призначена для прискорення процедури звернення до інших типів пам’яті, що входять до складу МПС. Ємність цієї пам’яті становить 64 кбайт – 1 Мбайт і постійно зростає у кожній наступній моделі комп’ютера. Швидкість роботи цієї пам’яті визначається швидкодією системної шини.

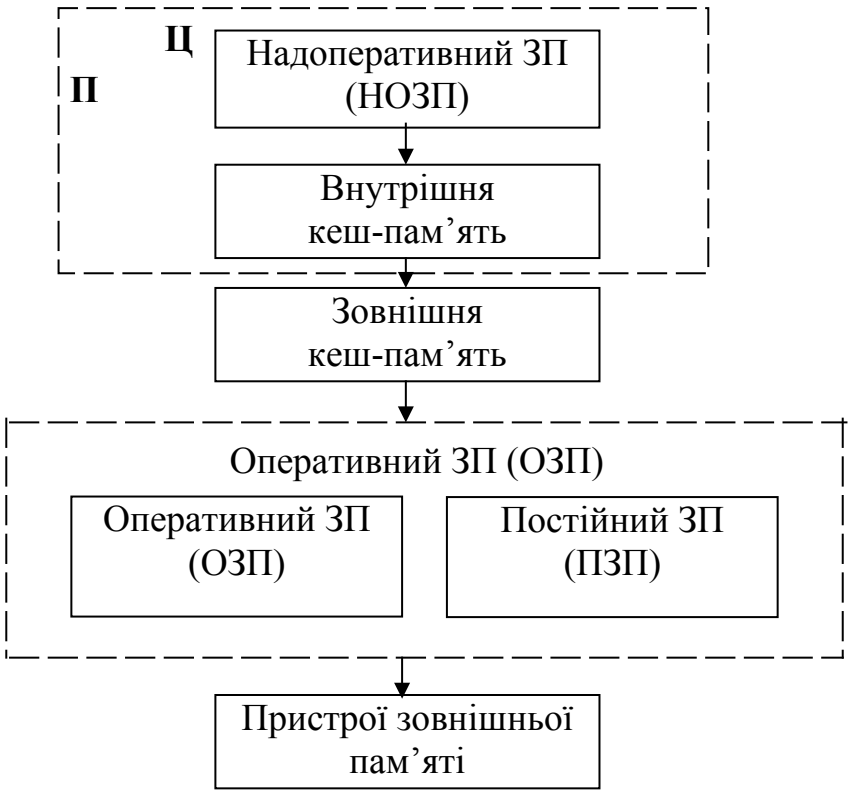


Рисунок 5.1 – Ієрархія пам’яті МПС

Оперативний запам’ятовувальний пристрій (ОЗП) – основний тип пам’яті МПС, який значною мірою визначає властивості МПС. Складається з двох видів пам’яті – **постійного запам’ятовувального пристрою** і, власне, **оперативного запам’ятовувального пристрою**. ОЗП будується за адресним принципом і має довільний доступ до кожної з комірок пам’яті.

Постійний запам’ятовувальний пристрій (ПЗП) – будується з мікросхем постійних (ПЗП) або перепрограмованих запам’ятовувальних пристроїв (ППЗП). Ці мікросхеми не допускають оперативної зміни інформації, що зберігається в них під час виконання програми, і не втрачають її при відключенні живлення. Інформація до них записується під час виготовлення або при першому програмуванні мікросхеми і надалі змінитися не може. Тому, під час роботи МПС інформація з них лише зчитується. У ПЗП зберігаються таблиці кодів команд, константи, стандартні підпрограми, наприклад, *BIOS* тощо. Ємність ПЗП складає 64...128 кбайт. Вартість біта інформації, що зберігається у ПЗП, може бути майже на порядок нижче ніж у ОЗП.

Оперативний запам’ятовувальний пристрій (ОЗП) – пристрій пам’яті призначений для записування, зберігання і зчитування будь-якої інформації під

час виконання програми. В залежності від типу використовуваних мікросхем, ОЗП буває двох видів: **статичний** і **динамічний**. Статичний ОЗП будується з мікросхем, елементом пам'яті яких є транзисторний тригер. Такий елемент пам'яті може зберігати інформацію дуже довго, доки є живлення, незалежно від кількості звернень для читання цієї інформації. ОЗП динамічного типу як елемент пам'яті використовує конденсатор, який є паразитною ємністю деяких схем, що побудовані з транзисторів зі структурою метал-діелектрик-напівпровідник (МДН). У результаті саморозряду такої ємності, інформація, що записана, може пропадати, тому вони потребують періодичного її поновлення (регенерування). При регенеруванні у кожний запам'ятовувальний елемент записується інверсія значення, що зберігалася до читання. В динамічному ПЗП є схема, що вказує, яке значення знаходиться у комірці пам'яті – пряме або інверсне.

До **пристроїв зовнішньої пам'яті** відносяться всі нагромаджувачі зі змінними і незмінними носіями: на твердих і гнучких магнітних дисках, лазерних компакт-дисках (CD-ROM) тощо. Обмін інформацією з такими пристроями відбувається з малою швидкістю, що пояснюється наявністю в їхньому складі електромеханічних вузлів. У цілому, ємність пристроїв зовнішньої пам'яті не обмежена. Можливо говорити лише про окремі пристрої, наприклад, ємність нагромаджувачів на твердих дисках становить 1...100 Гбайт.

Наявність того чи іншого типу пам'яті визначається функціональним призначенням МПС і умовами її роботи.

Крім адресної пам'яті до складу МПС можуть входити пристрої з іншим механізмом звернення до окремих комірок. До них відносяться асоціативна та стекова пам'ять.

Асоціативна пам'ять відрізняється тим, що звернення до комірок відбувається не за адресою, а за асоціативними ознаками самої інформації, що визначаються у результаті порівняння її з необхідними. До критеріїв визначення необхідної комірки можна віднести: рівність вмісту комірки наперед визначеному числу або будь-які інші. При цьому пошук за асоціативною ознакою (або послідовно за окремими розрядами) виконується паралельно у часі для всіх комірок масиву інформації, що зберігається. Це дозволяє підвищити швидкість оброблення інформації.

Стекова пам'ять (стек) – це пам'ять з послідовним доступом, звернення до комірок якої відбувається за безадресним принципом, за алгоритмом *LIFO (Last Input First Output)* – останній увійшов – перший вийшов. У МПС стекова пам'ять широко використовується під час виклику підпрограм, обробленні переривань та обробленні вкладених структур даних.

Стекову пам'ять реалізують за апаратним або апаратно-програмним способами.

При апаратній реалізації стекова пам'ять організована як сукупність регістрів, які зв'язані між собою так, що при зверненні до стека вміст його даних автоматично зсувається у той чи інший бік (залежно від операції, яка відбувається), чим забезпечується виконання алгоритму *LIFO*.

При апаратно-програмній реалізації стек організують в ОЗП статичного типу, а для визначення адреси існує спеціальний регістр *SP* (*Stack Pointer*) – **вказівник стека**, який вміщує адресу комірки пам'яті, з якої починається робота за алгоритмом *LIFO*. Ця комірка пам'яті називається **вершиною стека** і її вміст змінюється в залежності від виконання алгоритму *LIFO* для запису або читання даних у стеку.

Контрольні запитання:

1 Які параметри ЗП Ви знаєте?

2 Які параметри ЗП характеризують їх швидкодію?

3 Які принципи покладено в основу ієрархії ЗП МПС?

4 За якими принципами можливо здійснити звернення до певної комірки пам'яті у МПС?

Контрольні запитання підвищеної складності:

1 Чим можливо пояснити меншу швидкодію зовнішних пристроїв пам'яті?

2 За яким принципом відбувається звернення до комірок пам'яті у стеку?

3 Поясніть призначення вказівника стека.

5.2 Постійні запам'ятовувальні пристрої

Вхідний контроль:

1 Які технології виробництва мікросхем Ви знаєте?

2 Чим відрізняються одна від одної мікросхеми з різними технологіями?

ПЗП можуть бути зреалізовані при застосуванні різних фізичних принципів і елементів. Взагалі, вони відрізняються кількістю можливих змін інформації, способом запису інформації і способом її зміни. Напівпровідникові ПЗП є енергонезалежними пристроями з довільним вибором інформації.

За кількістю можливих змін інформації ПЗП поділяються на програмовані одноразово та багаторазово. В одноразово програмованих ПЗП інформацію записують один раз при виготовленні мікросхеми або перед першим включенням. У подальшому цю інформацію змінити неможливо. До таких мікросхем ПЗП відносяться: програмовані маскою ПЗП (ПЗПМ або *ROM– Read Only Memory*), інформація до яких записується шляхом створення на кристалі певного (замовного) фотошаблону, і мікросхеми програмовані одноразово (ППЗП – програмовані ПЗП або *PROM*), в яких користувач має змогу перепалити плавкі перемички на кристалі для занесення певної інформації.

ПЗП характеризуються найбільшою простотою організації та керування. ЕП напівпровідникових ПЗП – це звичайно один елемент (біполярний або МОН транзистор). Простота і малі розміри запам'ятовувальних елементів веде до збільшення щільності зберігання інформації, що, в свою чергу, веде до

збільшення інформаційної ємності і зменшення питомої вартості зберігання інформації.

Більшість ПЗП має словникову організацію, яка дозволяє одночасно зчитувати вміст кількох ЕП (звичайно, один байт інформації), які розташовані у певній комірці пам'яті, за власною адресою.

До групи багаторазово програмованих ПЗП відносять мікросхеми, в яких інформацію, що зберігається, можливо стерти за допомогою ультрафіолетового випромінювання, яке надходить до кристалу через спеціальне віконце у корпусі, (*EPROM*) і мікросхеми з електричним стиранням (*EEPROM*). До ПЗП з електричним стиранням також відносять **флеш-пам'ять.**

Програмовані маскою ПЗП. Основним елементом **програмованих маскою ПЗП** (ПЗПМ) є матриця нагромаджувача, яка складається з масиву ЕП, кожен з яких розташовано на перехресті рядка і стовпчика. Елемент пам'яті ПЗПМ є резистивна або напівпровідникова (діодна, транзисторна) перемичка між рядком і стовпчиком. Відповідно до розробленої маски, у процесі виготовлення мікросхеми, перемички формують у тих точках матриці, де має бути записана логічна 1 і не формують, де має бути логічний 0.

Для керування зчитуванням інформації з матриці нагромаджувача використовують дешифратори рядків і стовпців, на які інформація надходить з формувача адреси, який підключено до адресної шини МПС. Дешифратор рядків формує сигнал дозволу читання певного рядка, а дешифратор стовпців формує сигнал читання вмісту певних ЕП рядка. Сигнал з виходу дешифратора стовпців надходить на входи селектора, який з'єднує виходи вибраних ЕП з входами підсилювачів-формувачів вихідного сигналу. Спрощена структурна схема типового ПЗПМ показана на рис. 5.2. На цьому рисунку резистори, що з'єднують лінію вибраного рядка з лініями, які підключені до селектора, відповідають сигналам логічної 1, що записані у відповідні комірки. Нагромаджувач на цьому рисунку – це квадратна матриця $(N-1) \times (N-1)$ ЕП, з вмісту рядка якої можливо сформувати вихідні дані потрібного розміру.

Мікросхеми ПЗПМ мають багато варіантів вмісту інформації, що в них зберігається, і кількість їх модифікацій безперервно збільшується, тому за необхідності можна знайти необхідну.

Виходи усіх мікросхем ПЗПМ мають ТТЛ-рівні і вони побудовані, переважно, за схемою з трьома стійкими станами.

Одноразово програмовані ПЗП (ППЗП або *PROM – Programmed Read Only Memory*). Мікросхеми цього типу за принципами побудови і принципами функціонування аналогічні ПЗПМ, але допускають програмування їх безпосередньо користувачем. Побудування мікросхеми ППЗУ, в цілому, аналогічно поданому на рис. 5.2. Відміна полягає у наявності пристроїв формування струму програмування, який подається на вихідні виводи мікросхеми. Операція програмування відбувається шляхом знищення (перепалювання) плавких перемичок на поверхні кристалу в містах перехрестя рядків і стовпців матриці нагромаджувача, де потрібно записати логічний 0 або 1, тому ці мікросхеми програмуються лише один раз. В залежності від характеристик мікросхеми, ця матриця в начальному стані має вміст, що

складається з нулів або одиниць. Склад вмісту обумовлено характеристиками підсилювача зчитування, який може бути інвертором або повторювачем. Програмування мікросхеми, матриця якої в початковому стані заповнена 0, полягає в тому, що перепалюються перемички у тих місцях, де потрібно зберігати 1. Якщо в початковому стані матриця заповнена 1, то перепалюються перемички, де необхідно зберігати 0.

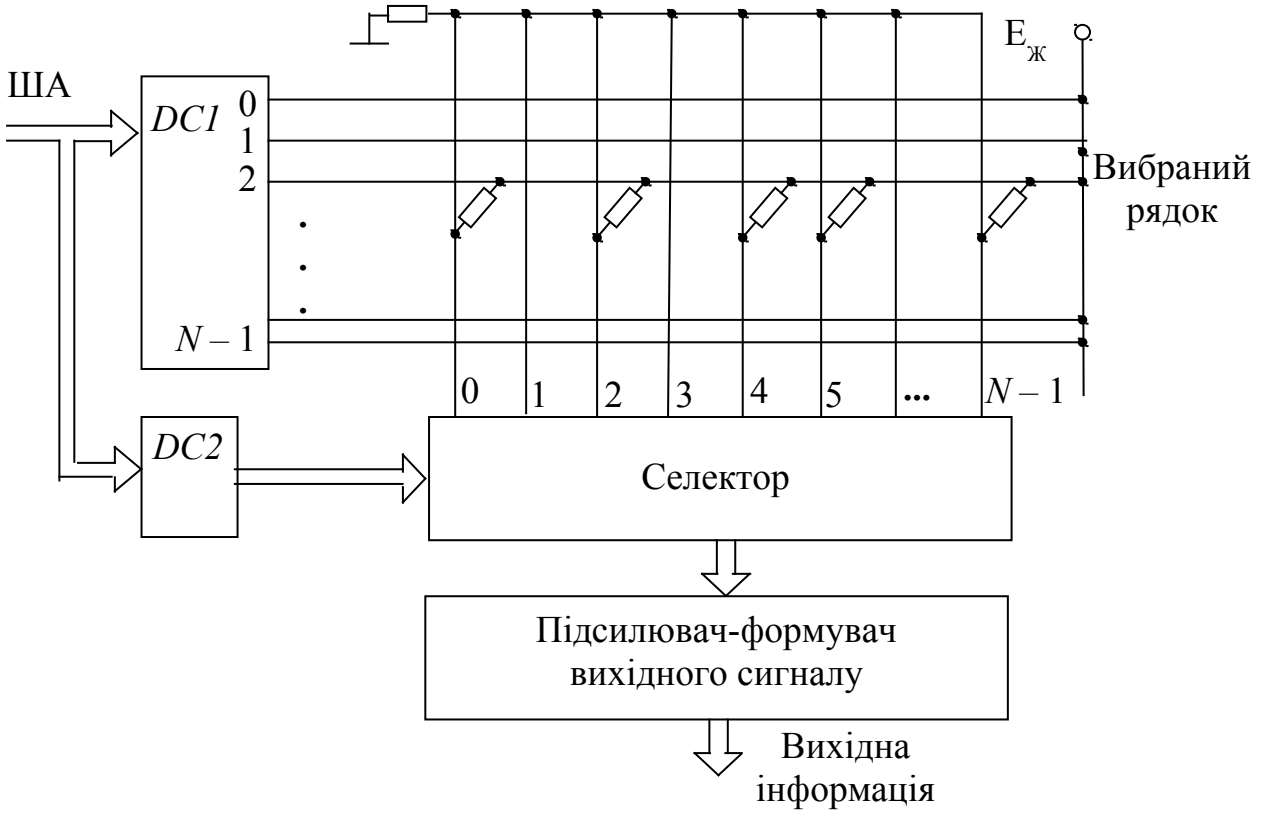


Рисунок 5.2 – Спрощена структурна схема ПЗПМ

Програмування проводиться за допомогою спеціальних пристроїв – програматорів, які є досить простими приладами. Програмування проводиться подаванням імпульсів електричного струму з амплітудою 30...50 мА, за відповідними адресами.

ППЗП використовуються для зберігання налагоджених програм для керування МПС різного призначення.

Більшість мікросхем ППЗП побудована за ТТЛШ-технологією, але є невелика частина мікросхем, які побудовані за іншими технологіями – ЕСЛ, КМДН тощо. Мікросхеми різних технологій різняться за деякими параметрами, здебільшого, за швидкодією, споживаною потужністю, організацією пам’яті. Втім, усі мікросхеми ППЗП, крім виготовлених за ЕСЛ та *n*-МДН технологіями, мають вихідні сигнали ТТЛ-рівнів, що забезпечує повну сумісність мікросхем різних типів.

ПЗП, багаторазово програмовані з ультрафіолетовим стиранням (репрограмовані ПЗП – РПЗП-УФ або *EPROM – Erasable Programmed Read Only Memory*). Ці мікросхеми дозволяють багаторазове їх перепрограмування

самим користувачем. Ця властивість забезпечується використанням *n*-МОН транзисторів з застосуванням механізму лавинної інжекції заряду (ЛІЗМОН) з подвійним затвором. Такі транзистори відрізняються від звичайних МДН транзисторів наявністю двопрошарового підзатворного діелектрика, який отримав назву «плаваючого затвору» (ПЗ). Прошарок діелектрика, який прилягає до каналу, виготовлено з окису кремнію, товщиною менше ніж 5 нм. Другий прошарок виготовлено з нітриду кремнію, товщиною близько 0,1 мкм. Електричний опір цього прошарку значно вище ніж опір прошарку окису.

Принцип роботи такого транзистора, який утворює елемент пам'яті, пов'язаний з нагромаджуванням заряду між прошарками діелектриків і впливом цього заряду на значення порогової напруги транзистора.

У режимі програмування на керувальний затвор, джерело і стік подають імпульс напруги 21…25 В позитивної полярності. У зворотно зміщених *p*-*n*-переходах виникає процес лавинного множення носіїв заряду й інжекції частини електронів у ПЗ. В результаті чого, на ПЗ нагромаджується негативний заряд, який зміщує передатну характеристику транзистора в область високої граничної напруги (праворуч), що відповідає запису 0 (рис. 5.3).

Для стирання інформації, перед перепрограмуванням, мікросхему розміщують під ультрафіолетове випромінювання дугової ртутної лампи. Під впливом цього випромінювання посилюється тепловий рух носіїв електричного заряду й електрони, що формували негативний заряд на ПЗ розсмоктуються у підкладку. Це зміщує передатну характеристику в область низької граничної напруги (ліворуч), що відповідає запису 1 (рис. 5.3).

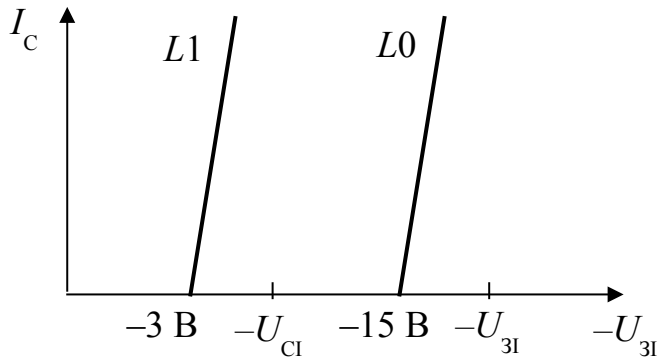


Рисунок 5.3 – Передатна характеристика *n*-МОН транзистора

Ці мікросхеми мають досить хороші властивості: порівняно високу швидкодію, значну кількість варіантів щодо організації пам'яті, невисоку вартість.

До недоліків цих мікросхем можна віднести: малу кількість циклів перепрограмування (від 10 до 100), що пояснюється швидким старінням діелектрика під впливом ультрафіолетового випромінювання, потребою у спеціальному обладнанні для стирання інформації, значний час стирання, досить високу чутливість до висвітлення та можливість випадкового стирання інформації.

ПЗП, багаторазово програмовані з електричним стиранням (репрограмовані ПЗП – РПЗП-ЕС або *EEPROM – Electrical Erasable Programmed Read Only Memory*). Ці мікросхеми за принципами побудування аналогічні РПЗП-УФ, але стирання інформації не потребує використання ультрафіолетового випромінювання. ЕП такої мікросхеми – це МОН-транзистор з індукованим каналом *p*-типу або *n*-типу, що має двопрошаровий діелектрик під затвором. Верхній прошарок формують з нітриду кремнію, нижній з – оксиду кремнію. Якщо до затвору відносно підкладки прикласти імпульс напруги позитивної полярності з амплітудою 30… 40 В, то під дією сильного електричного поля електрони проходять через прошарок оксиду кремнію і нагромаджують заряд між прошарками діелектриків. Цей заряд знижує граничну напругу і зміщує передатну характеристику транзистора ліворуч, що відповідає запису в ЕП логічної 1 (рис. 5.3). Для запису логічного 0 необхідно знищити заряд між шарами діелектриків, що нагромаджується при подачі на затвор імпульсу негативної полярності з амплітудою 30…40 В. При цьому заряд електронів витискається в підкладку. Відсутність заряду в прошарку діелектрика зміщує передатну характеристику в область високих граничних напруг (рис. 5.3).

Режими стирання і програмування забезпечуються напругами однієї полярності: негативної для *p*-МНОН структур і позитивної для *n*-МНОН структур.

Принципи побудови цих мікросхем аналогічні описаним вище. Крім вузлів, що забезпечують роботу мікросхеми як ПЗП, до їх складу входять пристрої, що забезпечують її роботу в режимах стирання і програмування – комутатори режимів і формувачі імпульсів необхідної амплітуди і тривалості з напруги програмування U_{PR} , що подається на відповідний вхід мікросхеми.

До переваг мікросхем РПЗП-ЕС можна віднести: значну кількість циклів перепрограмування і можливість її перепрограмування безпосередньо у складі певного пристрою, що розширює функціональні можливості таких мікросхем.

Флеш-пам’ять (*Flash*-пам’ять). Мікросхеми пам’яті такого типу були розроблені фірмою *Intel* у 1988 році.

Як ЕП флеш-пам’яті використовується МОН-транзистор з ПЗ, який виготовлено за спеціальною технологією, яка називається *ETOX* (*EPROM Thin Oxide*) і запатентована фірмою *Intel*. У цілому, структура МОН-транзистора з ПЗ подібна описаним вище. Відмінністю, яка забезпечуються технологією *ETOX,* є зменшення товщини прошарку оксиду кремнію більш ніж втричі, що дозволило зменшити напругу програмування до 12 В і зменшити напругу стирання за рахунок тунельного ефекту, також до 12 В. Ці заходи дозволяють виконувати перепрограмування флеш-пам’яті безпосередньо у складі МПС і забезпечують можливість збільшення кількості циклів запису інформації.

Для забезпечення правильної організації роботи флеш-пам’яті фірмою *Intel* розроблена низка заходів, що дозволяють уникнути виходу її з ладу під час програмування. До них можна віднести:

– застосування спеціальних алгоритмів запису і стирання з контролем стану і завершенням процесу за результатами контролю;

– попереднє програмування в режимі стирання, коли перед стиранням усі ЕП матриці установлюються в стан 0;

– включення до складу мікросхеми регістра, який зберігає ідентифікатори фірми-виробника й типу мікросхеми, що дозволяє захистити елемент від помилок вибору алгоритму;

– вбудування в мікросхему кіл, що реалізують алгоритм стирання і запису. Це спрощує зовнішнє керування і захищає від помилок під час перезапису.

Існує три групи мікросхем флеш-пам’яті:

– мікросхеми першого покоління, які виготовлені у вигляді єдиного масиву (блока), інформація в якому стирається цілком (*BULK-ERASE*);

– мікросхеми, масив пам’яті яких поділено на блоки різного розміру, що мають різні рівні захисту від випадкового звернення до них (*BOOT-BLOCK*);

– мікросхеми третього покоління, які мають найбільший розмір масиву, що поділено на блоки однакового розміру з незалежним стиранням (*FLASH-FILE*).

Мікросхеми різних груп мають відмінності в їх використанні. Так, мікросхеми *BULK-ERASE* можуть використовуватись замість традиційних мікросхем *EPROM*, з можливістю перепрограмування безпосередньо у складі обладнання під керівництвом процесора самої системи. Мікросхеми *BOOT-BLOCK* застосовуються для зберігання *BIOS* у персональних комп’ютерах, що дає можливість оновлення системи безпосередньо з зовнішніх носіїв інформації. Мікросхеми *FLASH-FILE* використовуються для зберігання даних великого обсягу в *Flash*-картах, які є альтернативою жорстким магнітним дискам. Очікується, що *Flash*-картки зможуть замінити жорсткі магнітні диски, особливо у системах, що працюють в умовах сильних механічних впливів.

Швидкодія флеш-пам’яті в 125…250 разів перевищує цей параметр для жорсткого диска, але поступається йому щодо інформаційної ємності, яка не перевищує 40 Мбайт.

Напруга живлення мікросхем флеш-пам’яті становить – 5 В, а стирання і програмування –12 В. Споживаний струм суттєво залежить від режиму роботи мікросхеми. Так, в режимі очікування (*Standby*) споживаний струм значно менший за струм, який споживається у режимі стирання і запису, переважно у колі джерела 12 В.

Більшість мікросхем флеш-пам’яті працюють з даними у вигляді послідовного коду з використанням шини I^2C (*Inter Integrated Circuit Bus*). Ця шина складається з двох двонаправлених ліній: *SCL* (*Serial Clock*) і *SDA* (*Serial Date*), до яких можна підключати до 128 пристроїв. Один з пристроїв є ведучим (*master*), інші – веденими (*slave*). Ведучий пристрій генерує імпульси синхронізації *SCL* і керує всією роботою шини. Ведені працюють під керуванням ведучого, обслуговуючи його запити.

Типова структурна схема мікросхеми флеш-пам’яті з послідовним введенням-виведенням інформації з використанням шини I^2C показана на рис. 5.4.

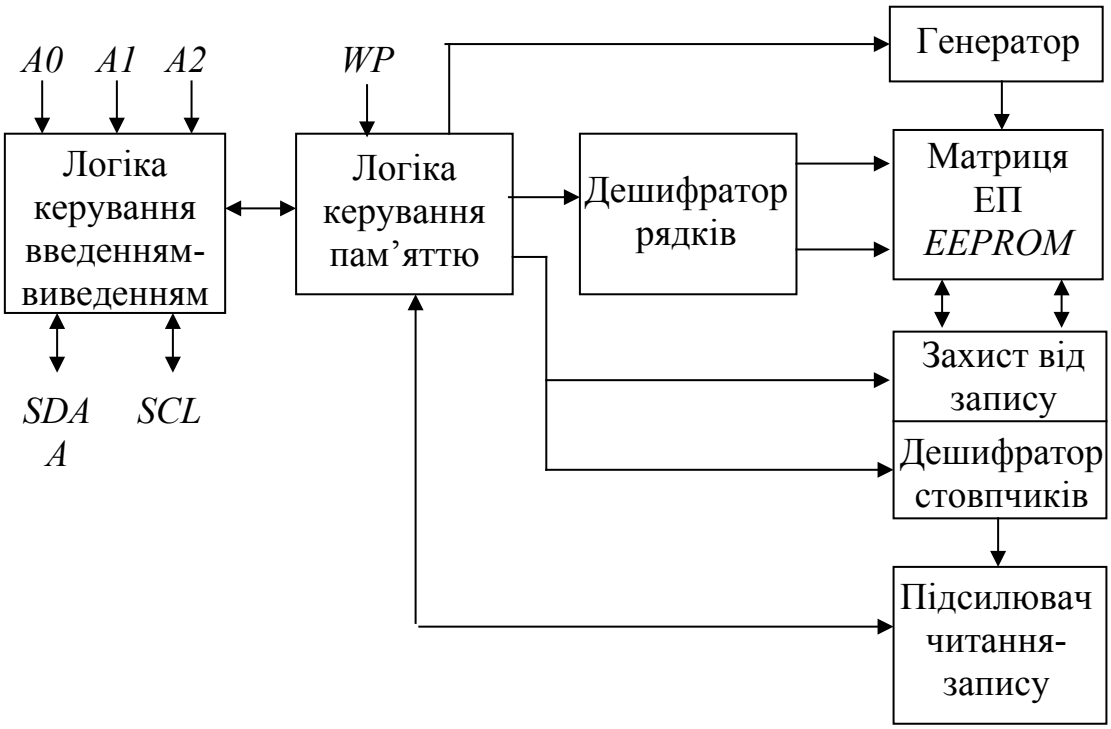


Рисунок 5.4 – Структурна схема ВІС флеш-пам'яті, що працює з шиною I^2C

На рис. 5.4 на блок логіки керування введенням-виведенням інформації надходять з шини I^2C сигнали $A0$, $A1$, $A2$, які визначають три молодші розряди адреси веденої ВІС, а старші чотири розряди адреси пам'яті не змінюються і мають значення 1010. По лініях *SCL* і *SDA* проводиться обмін відповідними сигналами. Вхід *WP* (*Write Protect*) блока логіки керування пам'яттю призначений для керування захистом даних, що записані у ВІС. Якщо цей вивід з'єднаний з загальним проводом, то можливо змінювати вміст будь-яких комірок пам'яті. Сигнал високого рівня, що подано на цей вивід захищає увесь масив даних або його частину від змін. Нині випускаються мікросхеми, в яких блокування стирання і запису можливо виконувати в інший спосіб, наприклад, за командами ведучого.

У неактивному стані шини I^2C на лініях *SCL* і *SDA* присутні високі рівні сигналу. Для початку сеансу роботи ведучий змінює стан лінії *SDA* на низький, не змінюючи стану лінії *SCL*. Після установлення низького рівня на лінії *SDA* змінюється стан лінії *SCL,* що відповідає команді СТАРТ. Передавання інформації відбувається побітно. Сеанс передавання закінчується командою СТОП, під час якої на лінії *SCL* установлюється високий рівень сигналу і за його наявності відбувається зміна стану лінії *SDA*. Часові діаграми процесу передавання інформації показано на рис. 5.5.

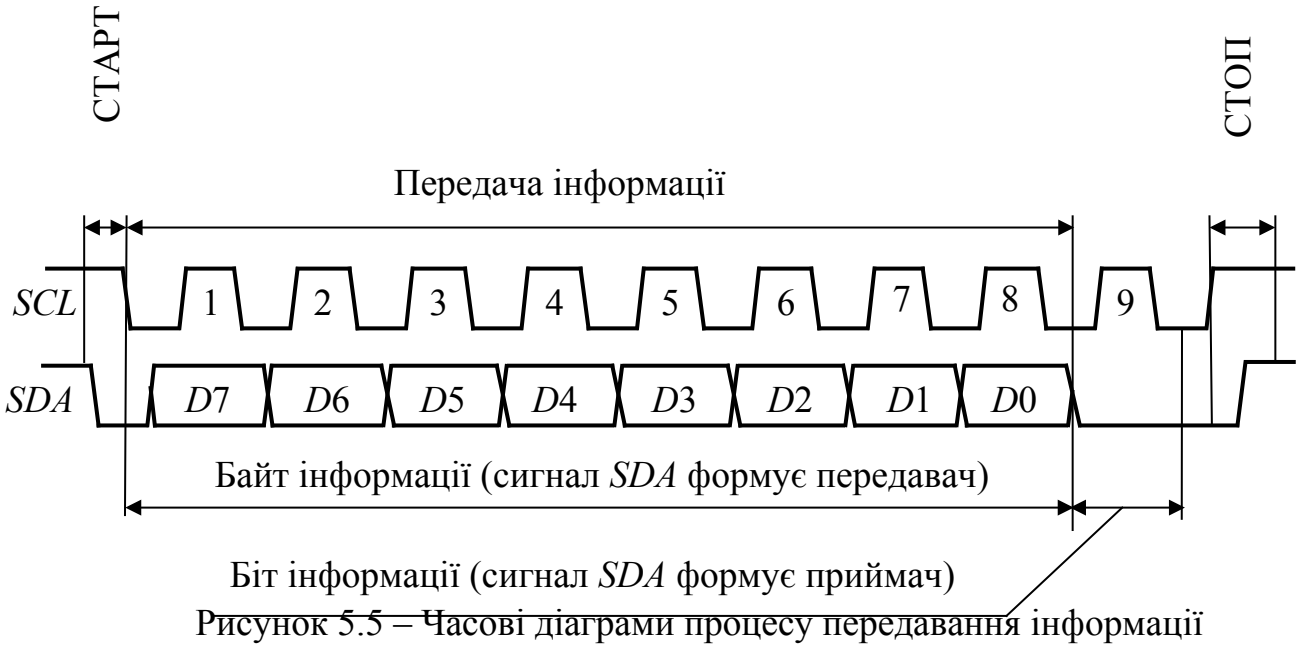


Рисунок 5.5 – Часові діаграми процесу передавання інформації

Частота проходження імпульсів синхронізації *SCL* становить 100 (400) кГц. При передаванні першим передається старший байт, а останнім – молодший. Пристрій, який прийняв байт, підтверджує його прийом, установлюючи на лінії *SDA* сигнал низького рівня, після чого формується сигнал СТОП.

Сім бітів, що передаються безпосередньо за командою СТАРТ, є адресою веденого пристрою, з яким необхідно установити зв’язок. Пристрій з такою адресою підтверджує приймання і готується до наступної роботи. Якщо у МПС немає пристрою з адресою, що передана по шині, то всі пристрої відключаються.

Молодший біт першого байта – це ознака напрямку передавання. Значення 0 цього біта відповідає напрямку від ведучого до веденого і не може змінитися у поточному сеансі роботи.

Контрольні запитання:

1 Як проводиться програмування ПЗП програмованих маскою?

2 Який пристрій використовується як елемент пам’яті у ПЗП з ультрафіолетовим стиранням?

3 Як відбувається робота флеш-пам’яті з шиною I2C?

Контрольні запитання підвищеної складності:

1 Які заходи застосовуються для уникнення виходу з ладу ВІС флеш-пам’яті?

2 Які типи ВІС флеш-пам’яті Ви знаєте?

3 Чим відрізняються ВІС РПЗП-УФ від ВІС РПЗП-ЕС?

5.3 Оперативні запам'ятовувальні пристрої

Вхідний контроль:

1 Які типи тригерів Ви знаєте?

2 Які характерні особливості мають тригери різних типів?

3 Сигнали, які необхідно використовувати для керування тригерами різних типів?

ОЗП статичного типу (*Static Random Access Memory – SRAM*) призначений для оперативного запису, зберігання і зчитування інформації під час виконання МПС будь-яких програм. У ОЗП статичного типу інформація зберігається у тому місці (комірці пам'яті або запам'ятовувальному елементі), де вона була записана і не руйнується під час її зчитування.

Структура ВІС ОЗП схожа на структуру мікросхем ПЗП з тією різницею, що як ЕП використовується транзисторний статичний тригер. Елементною базою для побудування тригерів можуть бути як біполярні, так і МОН-транзистори. Через те що, для функціонування тригера потрібне живлення, пам'ять такого типу є енергетично залежною (*volatile memory*). При відключенні живлення інформація, що зберігалася, втрачається. Запис інформації у тригер відбувається шляхом установлення в один з двох його можливих станів. Для зміни стану необхідно подати на входи тригера необхідні сигнали запису.

Типова структура ОЗП статичного типу включає: матрицю нагромаджувача і схеми запису-зчитування інформації, схеми дешифрування адреси ЕП або комірки пам'яті, схеми управління режимом тощо, які інтегровані на одному кристалі. У залежності від побудови нагромаджувача розрізняють ОЗП з однорозрядною та багаторозрядною організацією пам'яті. Основна відміна у структурних схемах полягає у тому, що нагромаджувач ОЗП з багаторозрядною організацією пам'яті складається з кількох прошарків однакових матриць і одну комірку пам'яті складають елементи з однаковими адресами в усіх матрицях.

Організація пам'яті однорозрядного ОЗП становить $2^m \times 1$ біт, де m – кількість розрядів шини адреси, що можуть бути підключені до цієї ВІС, а для багаторозрядного ОЗП становить $2^m \times n$ біт, де n – кількість розрядів шини даних. Багаторозрядні *SRAM*, переважно, мають байтову організацію ($2^m \times 8$).

Спрощену структурну схему багаторозрядного ОЗП з організацією $2^m \times 4$ показано на рис. 5.6.

В схемі, яка зображена на цьому рисунку, показано, що звернення проводиться одночасно до чотирьох матриць нагромаджувача по одному ЕП в кожній. Розряди шини адреси розподіляються на дешифратор рядків і стовпчиків для вибору відповідних рядків і стовпчиків одночасно в чотирьох матрицях.

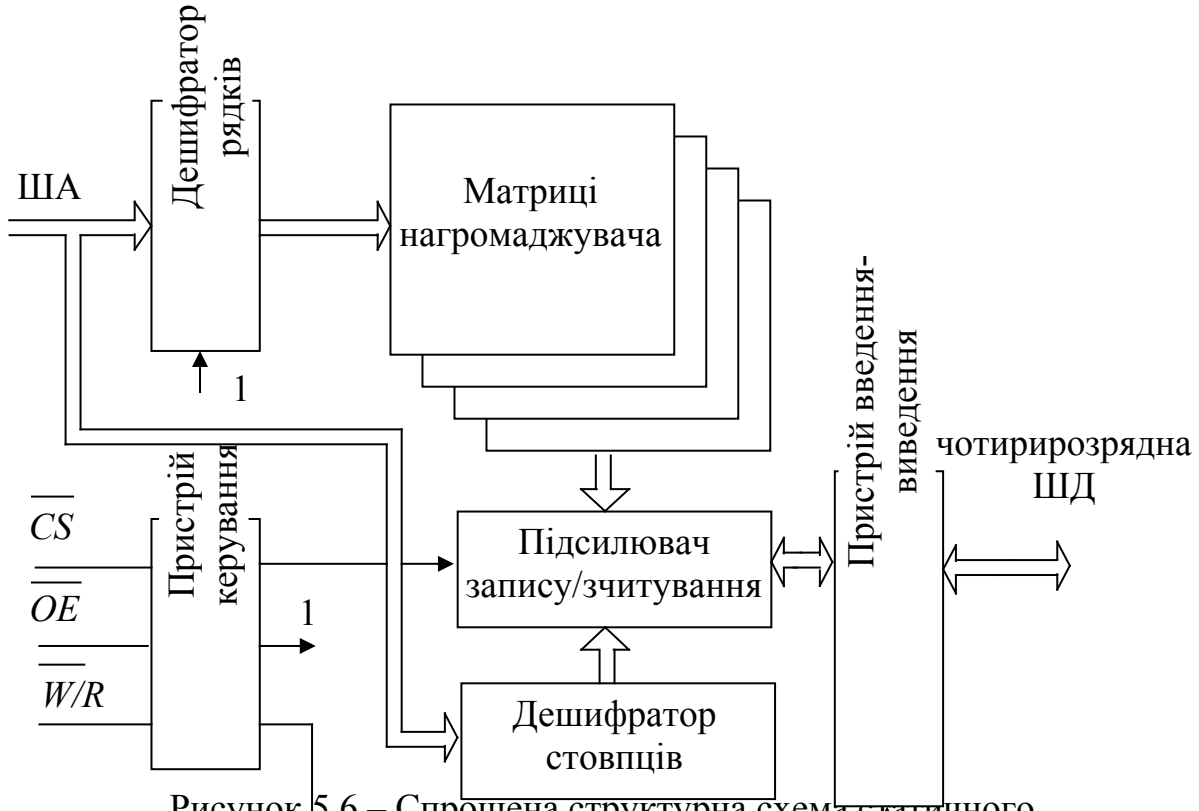


Рисунок 5.6 – Спрощена структурна схема статичного багаторозрядного ОЗП

На пристрій керування надходять такі сигнали:

– $\overline{CS}$ (*Chip Select*) – сигнал вибору мікросхеми. Сигнал логічного 0 на цьому виводі дозволяє роботу вибраної мікросхеми. Відсутність цього сигналу переводить мікросхему в неактивний стан;

▪

– $\overline{OE}$ (*Output Enable*) – сигнал дозволу на вихід. Сигнал логічного 0 (активний рівень для цього входу) дозволяє роботу виходу. Сигнал логічної 1 визначає перехід мікросхеми у *z*-стан;

▪

– $\overline{W/R}$ ($\overline{Write}$ / $\overline{Read}$) – $\overline{\text{запис}}$ / зчитування. Цей сигнал керує режимом роботи мікросхеми, забезпечуючи виконання необхідних функцій мікросхеми.

Для виконання операцій запису/зчитування необхідна одночасна наявність рівнів логічного 0 на виводах *CS* і *OE*. Відсутність будь-якого з них переводить мікросхему у режим зберігання інформації.

Запис інформації, яка надходить з чотирирозрядної шини даних, виконується сигналом логічного 0 на вході *W/R* при активних рівнях сигналів *CS* і *OE*. Запис проводиться у комірку пам’яті, адреса якої установлена на шині адреси.

Для зчитування вмісту комірки пам’яті необхідно подати активні рівні сигналів *CS* і *OE*, на шині адреси установити адресу необхідної комірки, на вхід *W/R* подати сигнал з рівнем логічної 1. Зчитування відбувається на чотирирозрядну шину даних.

Будь-які інші комбінації сигналів на входах керування переводять ВІС у режим зберігання інформації.

Сучасні мікросхеми *SRAM* мають інформаційну ємність до 36 Мбіт, а час вибірки менш ніж 5 нс.

ОЗП динамічного типу (*Dynamic Random Access Memory – DRAM*) також призначений для оперативного запису, зберігання і зчитування інформації під час виконання МПС будь-яких програм. Модулі ОЗП сучасних МПС, як правило, будуються на базі мікросхем такого типу.

В якості ЕП ВІС *DRAM* використовується ємність *p-n*-переходу МДН-транзистора, стан заряду якої відповідає інформації, що зберігається у цій комірці. Вважають, що заряджений конденсатор зберігає інформацію логічної 1, а розряджений – логічного 0. Для тривалого зберігання інформації виконується порядкова регенерація (*refresh*) всього вмісту *DRAM* з інтервалом 2 або 4 мс. Поновлення інформації відбувається також під час запису і зчитування інформації, а також під час спеціального циклу регенерації. Порівняно з ВІС *SRAM* мікросхеми ОЗП динамічного типу мають більшу інформаційну ємність. Останнім часом випускаються ВІС з організацією пам'яті 1М × 1, 4М × 1, 16М × 1, 64М × 1. До недоліків мікросхем *DRAM* можливо віднести лише меншу швидкодію.

Для забезпечення збільшення інформаційної ємності, мікросхеми *DRAM* повинні мати адресну шину з більшою кількістю розрядів, що викликає певні труднощі, тому всі ВІС цього типу мають мультиплексовану адресну шину. Звернення до ЕП відбувається за два етапи формування її адреси – окремо для рядка і окремо для стовпчика, що забезпечується наявністю двох спеціальних входів: *CAS* (*Column Address Strobe*) – строб адреси стовпця і *RAS* (*Row Address Strobe*) – строб адреси рядка.

Для запису або зчитування інформації з такої ВІС на адресну шину установлюють код адреси рядків (молодшу частину адреси) і на вхід *RAS* подають активний рівень сигналу (звичайно – логічний 0), який фіксує цю адресу у внутрішній регістр адреси рядків. Після чого, формується потрібний сигнал запису або зчитування. На адресній шині установлюється код адреси стовпчика (старша частина адреси) і на вхід *CAS* подається активний рівень сигналу. Негативний перепад проводить запис інформації у певну комірку. Зчитування інформації здійснюється негативним рівнем сигналу *CAS* після формування адреси стовпчика. Запис інформації в ЕП проводиться з вхідної лінії *DI* (*Date Input*). Часові діаграми процесу запису інформації показано на рис. 5.7.

Після закінчення процесу запису стан внутрішніх кіл ВІС необхідно відновити, подавши на вхід *RAS* високий рівень сигналу. Тривалість дії цього сигналу дорівнює інтервалу між сусідніми сигналами *RAS*.

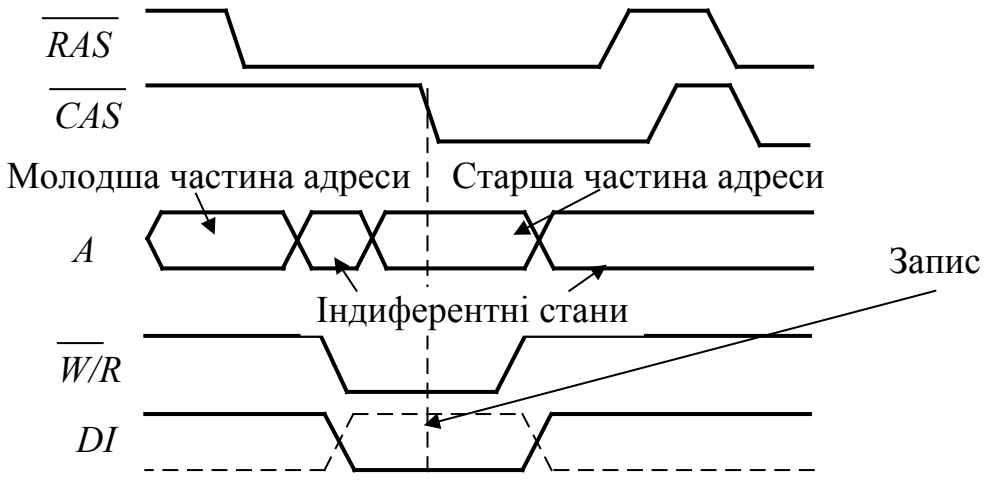


Рисунок 5.7 – Часові діаграми процесу запису інформації

Зчитування інформації проводиться на вихідну лінію *DO* (*Date Output*). Затримку зчитування вихідного сигналу можна відраховувати від негативного перепаду сигналу *CAS*. Процес зчитування показано на рис. 5.8.

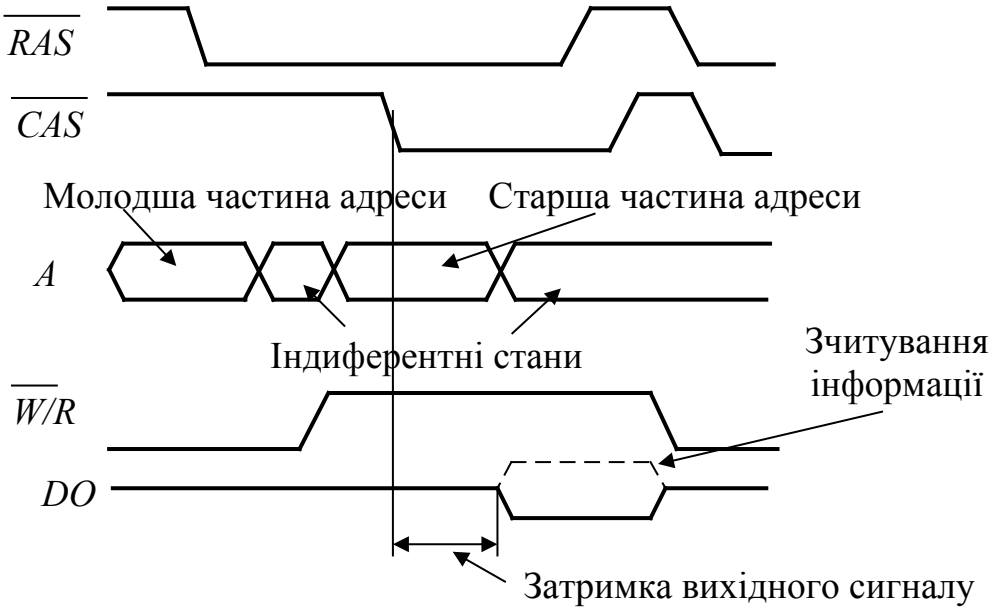


Рисунок 5.8 – Часові діаграми процесу зчитування інформації

Процес регенерації автоматично виконується для всіх ЕП рядка до якого відбувається звернення для запису або зчитування.

Цикл регенерації складається з послідовного перебору адрес усіх рядків і звернення до них. Формування адрес відбувається за допомогою зовнішнього лічильника циклів звернень. Звернення до матриці можливо організувати у кожному з можливих режимів функціонування: запису, зчитування, зчитування/модифікації/запису, а також у спеціальному режимі регенерації – сигналом *RAS* (за наявності сигналу *CAS,* що має неактивний рівень). Такий вид

регенерації називається прихованою регенерацією (*hidden refresh*), «прозорою» (*transparent refresh*) або захопленням циклу (*cycle stealing*).

ВІС типу *DRAM* можуть працювати з пам'яттю, що має сторінкову організацію. Сторінкові режими звернення до ВІС *DRAM* зреалізовано з вибіркою вмісту ЕП усього рядка при зміні адреси стовпчиків. У такому режимі зменшується час циклу звернення, тому що звернення до наступного байта відбувається без станів очікування і зміни лише частини адреси.

Для використання в складі комп'ютерів на базі процесорів 80386, *i*486, а також перших моделей *Pentium* використовувалися модулі пам'яті *SIMM* (*Single Inline Memory Modules* – модулі пам'яті з однорядковим розташуванням виводів). Оперативна пам'ять стандарту *SIMM* випускалася у двох модифікаціях: *FPM* (*Fast Page Mode*) з напругою живлення 5 В для комп'ютерів стандарту *IBM PC* 486 і більш сучасний варіант *EDO* (*Extended Data Output* – розширений вивід даних) з живленням 3,3 В. Перші модулі мали 30 виводів і організацію 1М × 8, 1М × 9, 4М × 8 і 4М × 9. Дев'ятий біт – біт контролю парності. Більш сучасні модулі *SIMM* мають 72 виводи й організацію – 1М × 32, 1М × 36 (з контролем на парність). Також є модулі з організацією 2М × 32, 4М × 32, 8М × 32, 16М × 32 тощо.

Зараз стандартними для більшості систем є модулі модифікації *DIMM DDR* (*Dual Inline Memory Modules Double Data Rate* – модулі пам'яті з дворядним розташуванням виводів і подвійним стандартом даних), які мають ємність від 64 М до 1 Гбайта. Існує декілька варіантів реалізації оперативної пам'яті стандарту *DIMM DDR*, що відрізняються пропускною здатністю, яка визначається кількістю біт за секунду, що приймаються і передаються оперативною пам'яттю в процесі її функціонування. Сьогодні випускаються модулі пам'яті *DIMM DDR* стандартів PC1600, PC2100, PC2700 і PC3200 (пропускна здатність 1600, 2100, 2700 і 3200 Мбайт/с відповідно).

Ще більш сучасними є модулі пам'яті *RIMM* корпорації *RAMBUS*, які мають більшу пропускну здатність ніж модулі *DIMM*. Пропускна здатність модуля *RIMM* на частоті 400/800 МГц становить 1,6/3,2 Гбайта/с.

Контрольні запитання:

1 В чому полягає різниця між статичними і динамічними ОЗП?

2 В чому полягає різниця між однорозрядними і багаторозрядними ОЗП?

3 Які сигнали керування забезпечують запис інформації в статичний ОЗП?

4 Яким чином досягнуто збільшення інформаційної ємності динамічних ОЗП?

5 Яке призначення сигналів *CAS* і *RAS* динамічного ОЗП?

Контрольні запитання підвищеної складності:

1 Які модифікації модулів *SIMM* Ви знаєте?

2 В чому полягає сутність процесу регенерування динамічної пам'яті?

5.4 Умовне позначення мікросхем пам’яті

Вхідний контроль:

1 Яке умовне графічне позначення використовується для зображення цифрових елементів на принципових схемах?

2 Які поля можуть бути на умовному графічному позначенні цифрових елементів і для чого вони використовуються?

Умовне позначення мікросхем пам’яті складається з трьох полів: основного – де знаходиться інформація про функціональне призначення мікросхеми й двох доповнювальних, в яких розміщується інформація про функціональне призначення кожного виводу й тип сигналів на входах даних. За необхідності, доповнювальні поля можуть поділятись на групи однакових, за функціональною ознакою, виводів – адресні, керувальні, інформаційні. Розміри кожного з полів установлюються відповідно до ДСТУ на зображення інтегральних мікросхем.

У залежності від функціонального призначення в основному полі, можливі такі позначення:

– *ROM* – загальне позначення постійного запам’ятовувального пристрою, без уточнення до якого типу відноситься. Якщо необхідно точно вказати використовуваний тип пристрою, то загальне позначення може доповнюватись:

– *PROM* – постійний запам’ятовувальний пристрій, програмований маскою;

– *EPROM* – репрограмований ПЗП з електричним записом і ультрафіолетовим стиранням;

– *EEPROM* – репрограмований ПЗП з електричним записом і електричним стиранням;

– *RAM* – умовне позначення оперативного запам’ятовувального пристрою статичного типу;

– *RAMD* – умовне позначення оперативного запам’ятовувального пристрою динамічного типу;

– *CAM* – умовне позначення асоціативного запам’ятовувального пристрою.

У лівому доповнювальному полі розміщується інформація про функціональне призначення входів, а у правому – виходів мікросхеми. Для різних типів ЗП виводи мають різне функціональне призначення, але в цілому, є певний набір, з якого можна вибирати необхідні:

A_0 –A_N – адресні входи, номер розряду адреси показано у вигляді індексу;

DI_0 – DI_M – входи даних. Для багаторозрядних ОЗП номер розряду вхідного сигналу подається у вигляді індексу;

DO_0 – DO_M – виходи даних. Для багаторозрядних ЗП номер розряду вхідного сигналу подається у вигляді індексу;

DIO_0 – DIO_M – входи/виходи даних. Виводи, функції яких об’єднано і вибір необхідної визначається сигналами керування;

C (*CLK*) – вхід синхронізації. Призначено для прийому сигналу синхронізації від генератора тактових імпульсів;

CAS – строб адреси стовпця;

RAS – строб адреси рядка;

EO – сигнал дозволу виходу;

ER (*Enable Read*) – сигнал дозволу стирання (обнулення вмісту ОЗП);

RD (*Read*) – сигнал дозволу зчитування;

WR (*Write*) – сигнал дозволу запису;

W/R – сигнал керування процесом запису/читання;

REF (*refresh*) – сигнал зовнішньої регенерації;

CS – сигнал вибору мікросхеми;

XACK – вхід сигналу кінця циклу запису/читання. Указує на закінчення циклу взаємодії динамічної пам'яті з центральним процесором. Формується контролером динамічної пам'яті;

SACK – вхід сигналу кінця циклу запису/читання. Указує на закінчення циклу взаємодії динамічної пам'яті з центральним процесором. Формується контролером динамічної пам'яті;

U_{PK} – вхід напруги програмування;

U_{CC} – вивід для подання напруги живлення;

GND – загальна лінія (цифрова «земля»).

Виводи U_{PK}, U_{CC} і *GND* на принципових схемах можна не показувати.

У залежності від режиму роботи й можливостей формувати різні сигнали, виходи мікросхеми можуть перебувати у трьох станах – стані формування логічного 0, стан формування логічної 1 й у високоімпедансному стані (*z*-стані). Високоімпедансний стан відповідає стану виходу, який відключено від лінії. Крім того, в залежності від схемотехнічних особливостей вихідних каскадів мікросхем, виходи можуть бути: по-перше – з відкритим колектором (відкритим стоком) – це вихідний каскад на біполярному або МОН-транзисторі в якого немає резистора у колі колектора або у колі стоку; по-друге – з відкритим емітером – це каскад на біполярному транзисторі в якого відсутній резистор у колі емітера. Для забезпечення правильної роботи таких схем необхідно передбачити підмикання резистора необхідної величини. Наявність у мікросхемі таких особливостей позначають спеціальними знаками:

 – вихід з трьома станами;

 – вихід з відкритим колектором;

 – вихід з відкритим емітером.

Приклади зображення мікросхем ПЗП різних типів подано на рис. 5.9.

На рис. 5.9,а зображена мікросхема ПЗП, програмована матрицею, ємністю 256 байт, з виходами, що можуть мати три стани. Сигнал вибору

мікросхеми *CS* має активний рівень, що дорівнює логічному 0. Тому для зчитування вмісту комірки необхідно подати код адреси і сигнал логічного 0 на вхід *CS*); на рис. 5.9,б показана мікросхема РПЗП, програмована однократно з організацією 256 × 4, вихідні каскади якої побудовано за схемою з відкритим колектором. Керування зчитуванням відбувається як і у попередньому випадку. Сигнал логічної 1 на вході *CS* переводить мікросхему у режим зберігання інформації; на рис. 5.9,в показана мікросхема РПЗП з електричним стиранням. Виводи мікросхеми двонаправлені. Вибір режиму роботи відбувається у відповідності з табл. 5.1.

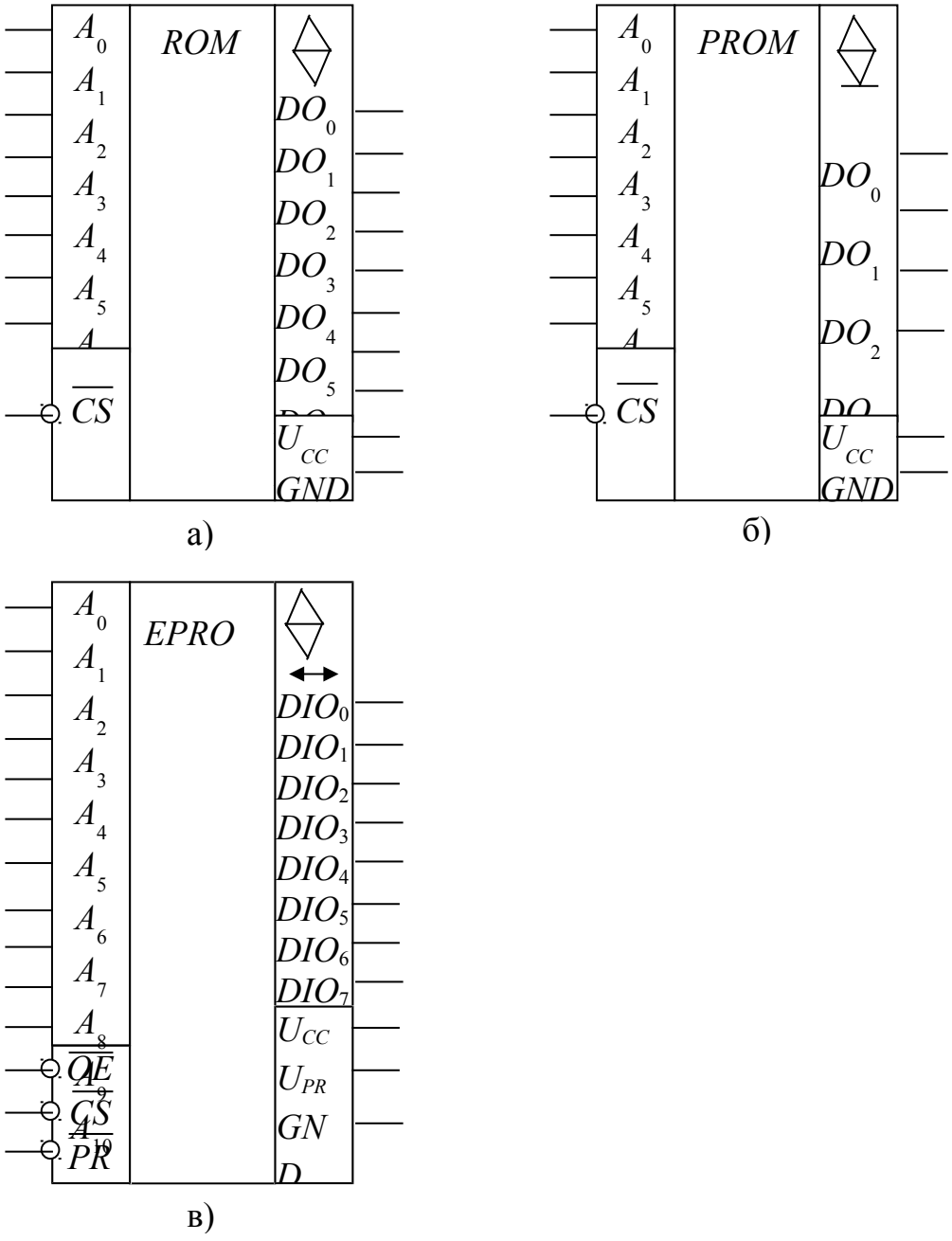


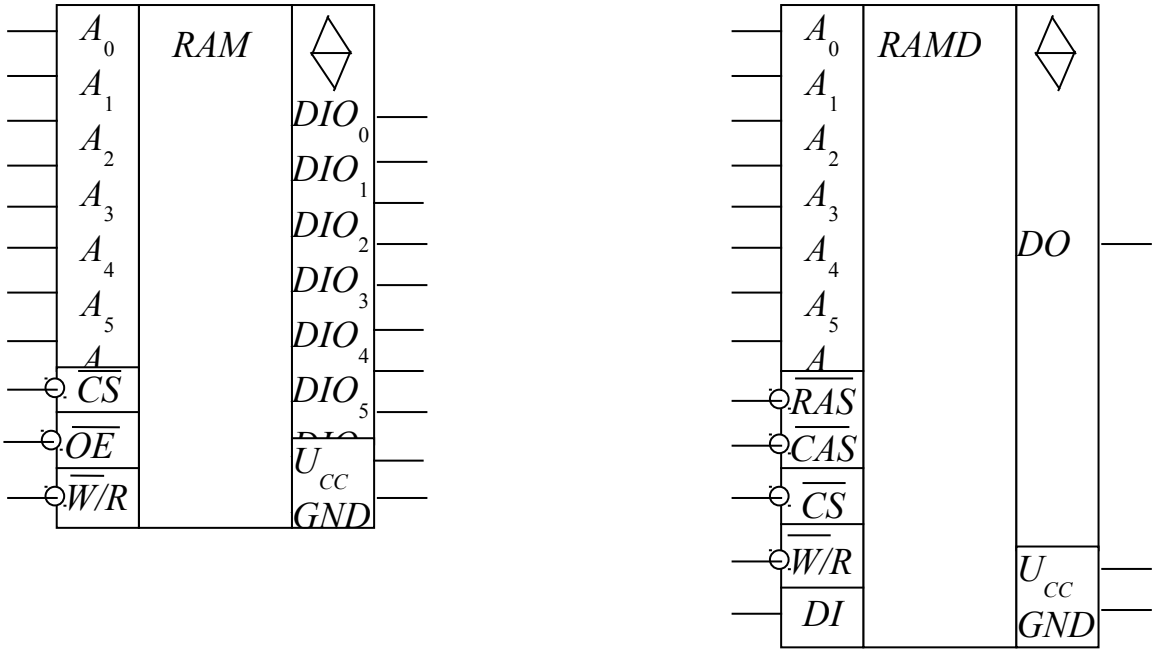
Рисунок 5.9 – Умовні графічні позначення мікросхем ПЗП

Таблиця 5.1 – Таблиця роботи мікросхеми РПЗП

Назва сигналу, значення сигналу					Режим роботи
CS	*EO*	$A_0 - A_{10}$	$DIO_0 - DIO_7$	U_{PR}	
1	X	X	*Z*	U_{cc}	Зберігання
1	0	X	0	U_{PR}	Стирання
1	1	*A*	$D_0 - D_7$	U_{PR}	Програмування
0	0	*A*	$D_0 - D_7$	U_{cc}	Зчитування

В таблиці літерами позначено: X індиферентний стан; A – десятирозрядний код адреси; U_{cc} – напруга живлення; U_{PR} – напруга переривання.

Приклади умовних графічних зображень ОЗП показано на рис. 5.10



а) б)

Рисунок 5.10 – Приклади умовних графічних позначень мікросхем ОЗП

На рис. 5.10,а зображено умовне графічне позначення мікросхеми ОЗП статичного типу з організацією 256 × 8. Керування режимом роботи здійснюється сигналами *OE, CS, W/R* відповідно до табл. 5.2.

На рис. 5.10,б зображена ОЗП динамічного типу з організацією 64К × 1, тому що її 8 входів даних адресують масив пам’яті 256 × 256 ЕП. Керування режимами роботи такої мікросхеми відбувається відповідно до табл. 5.3.

Таблиця 5.2 – Таблиця істинності мікросхеми ОЗП

Назва сигналу, значення сигналу					Режим роботи
CS	*OE*	*W/R*	A_0–A_7	DIO_0–DIO_7	
1	X	X	X	*Z*	Зберігання
0	X	0	*A*	0	Запис 0
0	X	0	*A*	1	Запис 1
0	1	1	*A*	*Z*	Зчитування без видачі
0	0	1	*A*	D_0–D_7	Зчитування

Таблиця 5.3 – Таблиця істинності мікросхеми ОЗП динамічного типу

Назва сигналу, значення сигналу						Режим роботи
RAS	*CAS*	*W/R*	*A*	*DI*	*DO*	
1	1	X	X	X	*Z*	Зберігання
1	0	X	X	X	*Z*	Зберігання
0	1	X	*A*	X	*Z*	Регенерація
0	0	0	*A*	0	*Z*	Запис 0
0	0	0	*A*	1	*Z*	Запис 1
0	0	1	*A*	X	*D*	Зчитування

Контрольні запитання:

1 Як позначається вивід ВІС з відкритим колектором?

2 Поясніть призначення сигналу *CS*?

3 В яких трьох станах може бути вихід ВІС пам’яті?

Контрольні запитання підвищеної складності:

1 Які умовні позначення використовуються для позначення типу вихідних ліній у ВІС?

2 Які виводи ВІС пам’яті призначено для прийому даних і як їх можливо організувати?

3 В яких трьох станах можуть знаходитися виводи введення-виведення даних до ВІС пам’яті?

4 Для яких цілей передбачено високоімпедансний стан виходу ВІС пам’яті?

5.5 Побудова модуля запам’ятовувального пристрою МПС з заданою організацією

Задача побудови модуля (блока) ЗП у залежності від завдання і початкових умов може вирішуватися у різний спосіб. Умови задачі також можуть мати свої відмінності в залежності від конкретних умов функціонування цього блока.

Технічне завдання на розробку модуля ЗП повинно містити:

– технічні характеристики МПС, для якої буде розроблятися модуль: потрібна інформаційна ємність, розрядність та типи сигналів шини адреси та шини даних, розподіл адресного простору, наявність сигналів керування та їхні рівні, довжина ліній проходження сигналів, інформаційна організація модуля пам'яті, наявність блоків живлення – рівні напруг та величини електричних струмів, які вони забезпечують, тощо;

– часові характеристики сигналів керування: тривалість імпульсів керування, часові затримки між ними, співвідношення між перепадами цих сигналів;

– часові характеристики сигналів, які формуються блоком, що розробляється, їх співвідношення з внутрішніми і зовнішніми сигналами в МПС ;

– електричні характеристики сигналів на виході блоку ЗП: рівні напруги, навантажувальна спроможність виходів блока.

Головна задача, що при цьому вирішується – забезпечення необхідної інформаційної ємності і забезпечення розрядності сигналів даних. Розв'язання цієї задачі, в залежності від наявності мікросхем пам'яті може бути двох видів:

– у номенклатурі мікросхем пам'яті існує ВІС, яка відповідає завданню на інформаційну організацію модуля і забезпечує відповідні часові характеристики. В цьому випадку мікросхема установлюється у МПС і виконується узгодження рівней сигналів керування або їх формування в разі необхідності. На цьому побудова модуля пам'яті вважається закінченою;

– у номенклатурі мікросхем пам'яті не існує ВІС, яка відповідає завданню на організацію модуля. В цьому випадку необхідно з наявних типів ВІС побудувати схему, що буде відповідати завданню. Ця задача має два варіанти: по-перше – у наявності є мікросхеми, що мають необхідну інформаційну ємність, але мають меншу розрядність даних; по-друге – у наявності можуть бути ВІС, які мають меншу інформаційну ємність, ніж задано, але забезпечують розрядність шини даних.

Побудова модулів пам'яті ПЗП і ОЗП відбувається аналогічно. Різниця між ними полягає лише у необхідності забезпечення і формування специфічних для кожного з них сигналів керування.

Розглянемо приклад побудови ПЗП для використання у МПС, яка має 24-розрядну шину адреси, 8-розрядну шину даних і на шині керування якої формуються сигнали:

– $\overline{OE}$ – з активним рівнем логічного 0, окремо для блока ПЗП;

– $\overline{W}/R$ – сигнал керування процесом запису-зчитування, активний рівень логічного 0 має сигнал *W*.

Інформаційна організація модуля, що розробляється, становить 115К × 8, в модулі пам'яті необхідно використовувати мікросхему РПЗП-УФ типу *AM*27*C*512, що має організацію 64К × 8. Початкова адреса комірки пам'яті для модуля – 000000_B.

По-перше, визначимо кількість мікросхем для забезпечення необхідної організації

$$N = \frac{115K \times 8}{64K \times 8} = 1{,}7968 \approx 2$$

Якщо в результаті отримано дробове число, то його необхідно **обов'язково** заокруглити до більшого цілого числа. По-друге, визначимо останню адресу комірки пам'яті модуля. Дві ВІС модуля повинні працювати по-черзі, обробляючи кожен по 64 К інформації. У сумі обидві ВІС можуть обробити 128 К інформації, що буде відповідати $2^{17} = 131072_D$ адресам. Якщо подати це число у шістнадцятьковій системі числення, то отримаємо число $1FFFF_H$, яке й буде останньою адресою модуля. Це число більше ніж задане – 115 К, це означає, що останні комірки пам'яті модуля ніколи використовуватись не будуть.

ВІС, яку необхідно використовувати має такі виводи: шістнадцять адресних входів – $A_0 – A_{15}$; вісім виходів даних – $DO_0 – DO_7$; входи керування – $\overline{OE}$, $\overline{CS}$ і $\overline{CE}$. Таким чином, необхідно з'єднати 2 ВІС з шинами МПС і між собою так, щоб забезпечити чергування їхньої роботи в залежності від установлення адреси.

Схема блока наведена на рис. 5.11.

Таблиця істинності мікросхеми *AM*27*C*512 наведена у табл. 5.4.

Таблиця 5.4 – Таблиця істинності мікросхеми *AM*27*C*512

Назва сигналу, значення сигналу					Режим роботи
CE	*OE*	*CS*	A_0–A_{15}	*DO*	
1	X	X	X	z	Неактивна
0	1	X	X	z	Неактивна
0	0	1	X	z	Неактивна
0	1	0	X	z	Зберігання
0	0	0	*A*	*D*	Читання

Робота кожної з мікросхем відбувається відповідно до цієї таблиці за сигналами керування, що надходять від МПС. Робота модуля відбувається в діапазоні адрес, який був визначений раніше. Якщо МПС формує адресу більшу ніж $01FFFF_H$, то на виході 7-вхідного елемента АБО (елемент *DD*1) формується сигнал логічної 1, який з'явиться на виході 2-вхідного елемента АБО (елемент *DD*2) і переведе обидві мікросхеми ВІС до неактивного стану.

Вибір однієї з двох мікросхем ПЗП відбувається сигналом A_{16} (лінія 17 адресної шини). Якщо на цій лінії є сигнал логічного 0 адреси в діапазоні $000000_H – 00FFFF_H$), то дозволяється робота мікросхеми *DD*4 на вхід *CE* якої надходить сигнал логічного 0. Якщо діапазон адрес, який виставлено на шину адреси становить $010000_H – 01FFFF_H$, то дозволяється зчитування з мікросхеми *DD*5 на вхід *CE* якої надходить сигнал логічного 0 з виходу інвертора (елемент *DD3*). Зчитування інформації відбувається сигналом $\overline{W}/R$, що надходить на входи *OE* обох мікросхем.

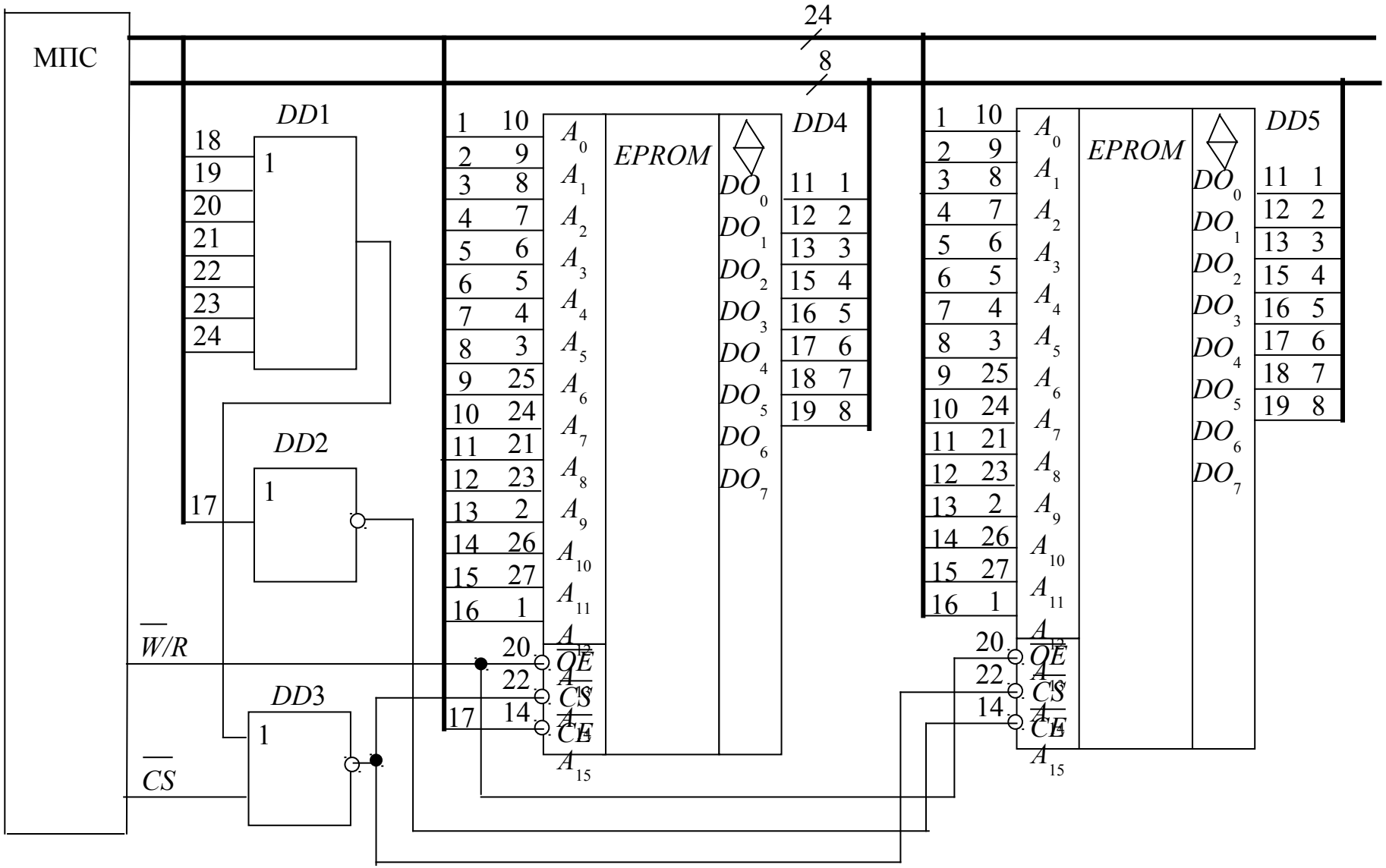


Рисунок 5.11 – Модуль ПЗП з організацією 128 К × 8

Розглянемо приклад побудування ПЗП для використання в МПС, яка має 24-розрядну шину адреси, 8-розрядну шину даних і на шині керування якої формуються сигнали:

- $\overline{OE}$ – з активним рівнем логічного 0, окремо для блока ПЗП;
- $\overline{W}/R$ – сигнал керування процесом запису-зчитування, активний рівень логічного 0 має сигнал W;

інформаційна організація модуля, що розроблюється становить 64 К × 16, але розрядність шини даних становить 16 розрядів (слово). У модулі пам'яті необхідно використовувати мікросхему ОПЗП статичного типу *AM*21*C*512, що має організацію 64 К × 8. Початкова адреса комірки пам'яті для модуля – 000000_B.

Визначимо кількість мікросхем для забезпечення необхідної організації

$$N = \frac{115K \times 16}{64K \times 8} = 1{,}7968 \approx 2$$

Якщо у результаті отримано дробове число, то його необхідно заокруглювати **обов'язково** до більшого цілого числа.

Визначимо останню адресу комірки пам'яті модуля. Тому що інформаційна ємність модуля складає 64 К, то остання адреса буде $00FFFF_H$.

ВІС, яку необхідно використовувати має такі виводи: шістнадцять адресних входів – A_0 – A_{15}; вісім входів/виходів даних – DIO_0 – DIO_7; входи керування – $\overline{OE}$, $\overline{CS}$, $\overline{CE}$ і $\overline{W}/R$.

Для збільшення кількості розрядів шини даних слід з'єднати мікросхеми таким чином, щоб забезпечити їх одночасну роботі з однаковими адресами.

Схема блока наведена на рис. 5.12.

Робота мікросхем ОЗП відбувається згідно з таблицею істинності (табл. 5.5) відповідно до сигналів керування, що надходять від МПС. Обидві мікросхеми працюють одночасно (паралельно з сигналами адрес і сигналами керування).

Таблиця 5.5 – Таблиця істинності мікросхеми ОЗП типу *AM*21*C*512

Назва сигналу, значення сигналу						Режим роботи
CE	*OE*	*CS*	*W/R*	A_0–A_{15}	DIO_0–DIO_8	
1	1	1	X	X	z	Неактивний стан
1	1	0	X	X	z	Неактивний стан
0	1	0	X	X	z	Зберігання
0	0	0	1	*A*	*DI*	Запис вхідних даних
0	0	0	0	*A*	*DO*	Зчитування даних

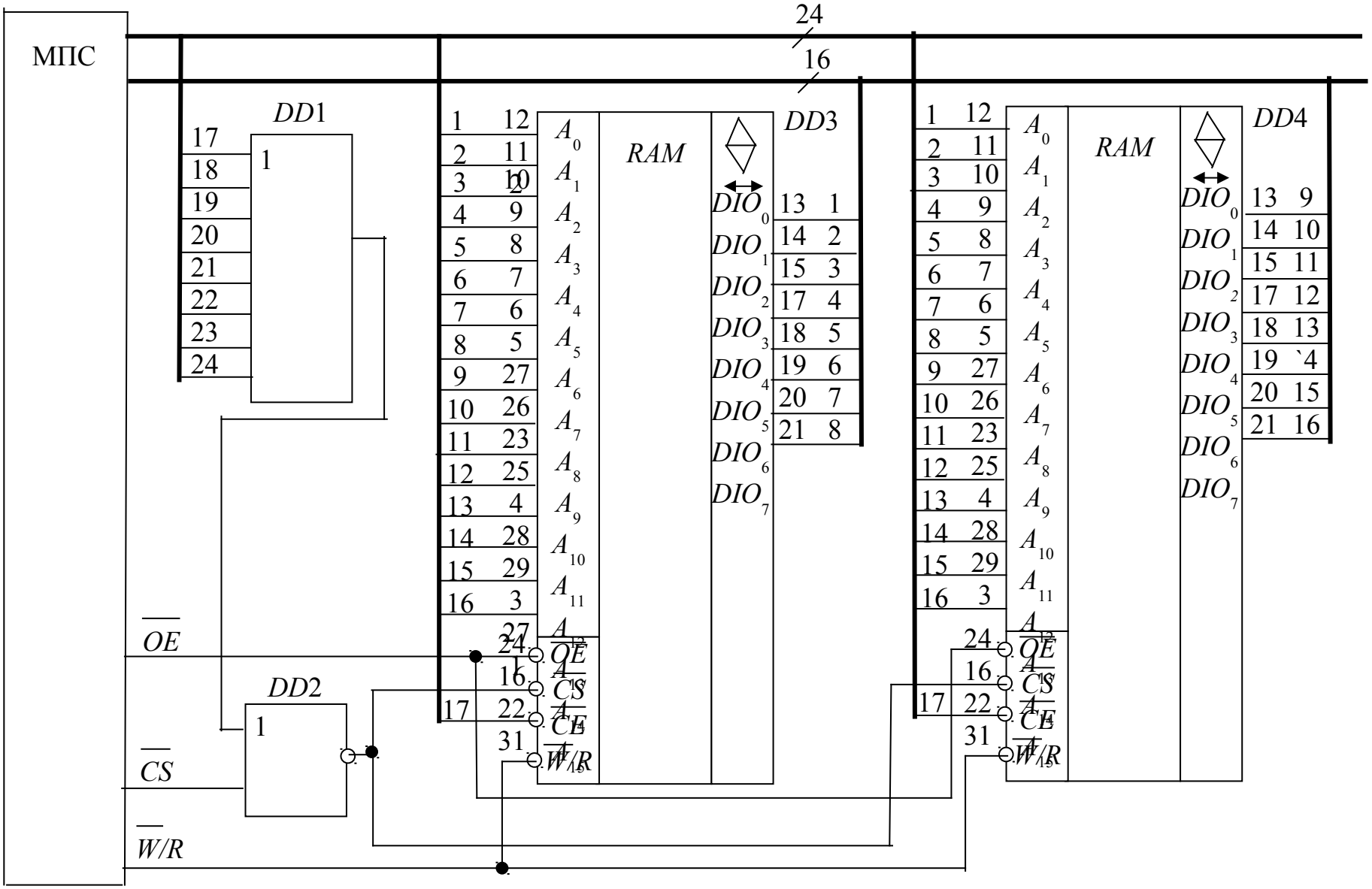


Рисунок 5.12 – Модуль ОЗП з організацією 64 К × 16

Контрольні запитання:

1 Які головні задачі вирішуються при побудуванні пристроїв ЗП?

2 Як визначається кількість мікросхем пам’яті, які необхідно використовувати?

3 Як відбувається нарощування інформаційної ємності ЗП?

4 Яким чином проводиться збільшення кількості розрядів виводів введення-виведення даних?

Контрольні запитання підвищеної складності:

1 Які типи ЗП входять до складу підсистеми пам’яті МПС?

2 В чому полягає адресний принцип звернення до підсистеми пам’яті МПС?

3 Скільки комірок пам’яті і скільки бітів інформації зберігається в цій комірці, якщо інформаційна організація модуля пам’яті становить 16 К × 8?

4 Чим відрізняється стекова пам’ять, що організована програмно, від ОЗП МПС?

5 Побудуйте схему модуля ПЗП з організацією 512 × 8 з мікросхем, умовне позначення яких подано на рис. 5.9,б.

6 ІНТЕРФЕЙС

6.1 Організація інтерфейсів

Вхідний контроль:

1 Які вузли ПК слугують забезпеченню взаємодії між мікропроцесором, пам’яттю та периферійними пристроями?

2 Що таке АЦП?

3 Що таке ЦАП?

Інтерфейсом називається система шин, допоміжної апаратури і алгоритмів, реалізованих апаратно, призначених для організації обміну між мікропроцесором, пам’яттю та пристроями введення-виведення. До функцій інтерфейсу входить дешифрування адреси пристрою пам’яті або введення-виведення, а також зовнішнього пристрою, синхронізації обміну інформацією, узгодження форматів слів, електричне узгодження сигналів різних підсистем та вузлів МПС.

Складність задач, вирішуваних інтерфейсом, часто недостатня потужність буферних схем, які входять до складу самої ВІС МП, призводять до розподілення засобів інтерфейсу між різними пристроями:

1 Пристроями керування пам’яттю та введення-виведення.

2 Безпосередньо інтерфейсним пристроєм, який є проміжним ланцюгом між МП і пам’яттю та пристроями введення-виведення (системний інтерфейс).

3 Спеціалізованими пристроями керування (контролерами) введенням-виведенням, призначеними для реалізації алгоритмів керування, специфічних для різних пристроїв (клавіатури, магнітних дисків, моніторів тощо).

4 Контролерами для спряження за допомогою шин з периферійними пристроями (датчиками, виконавчими механізмами, комутаторами, АЦП, ЦАП).

Організація обміну між МП та пам’яттю або введенням-виведенням у простих випадках можлива на основі засобів, які є у самому МП. Ті функції, які не підтримуються апаратно мікропроцесором, реалізуються програмно.

Більш складні запам’ятовувальні пристрої та пристрої введення-виведення підключаються до МП через додаткові інтерфейсні пристрої, контролери, які виконуються у вигляді спеціальних ВІС, що входять до мікропроцесорного комплекту.

Керувальні сигнали МП повинні забезпечувати функціональну та часову взаємодію між МП, пам’яттю та введенням-виведенням у процесі обміну інформацією в основних режимах:

– для організації обміну з пам’яттю, крім адресних сигналів та сигналів даних повинні видаватись сигнали запису, читання, іноді сприйматися сигнал готовності ЗП;

– для програмного обміну даними, крім адресної інформації та самих даних, необхідні код вибирання пристрою введення-виведення, сигнали

керування записом та читанням, повинні сприйматися сигнали готовності введення-виведення тощо;

– для організації прямого доступу до пам’яті (ПДП) повинні бути підключені шина адреси для передавання адреси пам’яті у контролер ПДП та керувальні сигнали: код вибирання пристрою введення-виведення, запити від цього пристрою, дозвіл на обмін даними;

– у процесі роботи МП виробляє сигнали стану (останов, чекання роботи у ПДП тощо) та синхронізуючі сигнали для інших пристроїв;

– для обміну даними у режимі переривань необхідно видавати з мікропроцесора керувальні сигнали дозволу переривань, підтвердження переривань та отримувати при готовності пристрою введення-виведення сигнали запиту на переривання.

Область використання зумовлює кількість та склад потрібних пристроїв введення-виведення і каналів зв’язку в МПС. Відповідно до цього у підсистемі введення-виведення можна виділити два рівні спряження підсистеми з процесором та пам’яттю. На першому рівні пристрої введення-виведення самі або через контролери поєднуються з процесором і пам’яттю за допомогою системного інтерфейсу МПС, який об’єднує окремі підсистеми в єдину систему. На другому рівні спряження контролери через канали зв’язку поєднуються з відповідними зовнішніми, або периферійними, пристроями (ПП) мікропроцесорної системи.

Незважаючи на широке застосування системних інтерфейсів, можна виділити два способи звернення до пристроїв введення-виведення:

– з застосуванням спеціальних команд введення-виведення;

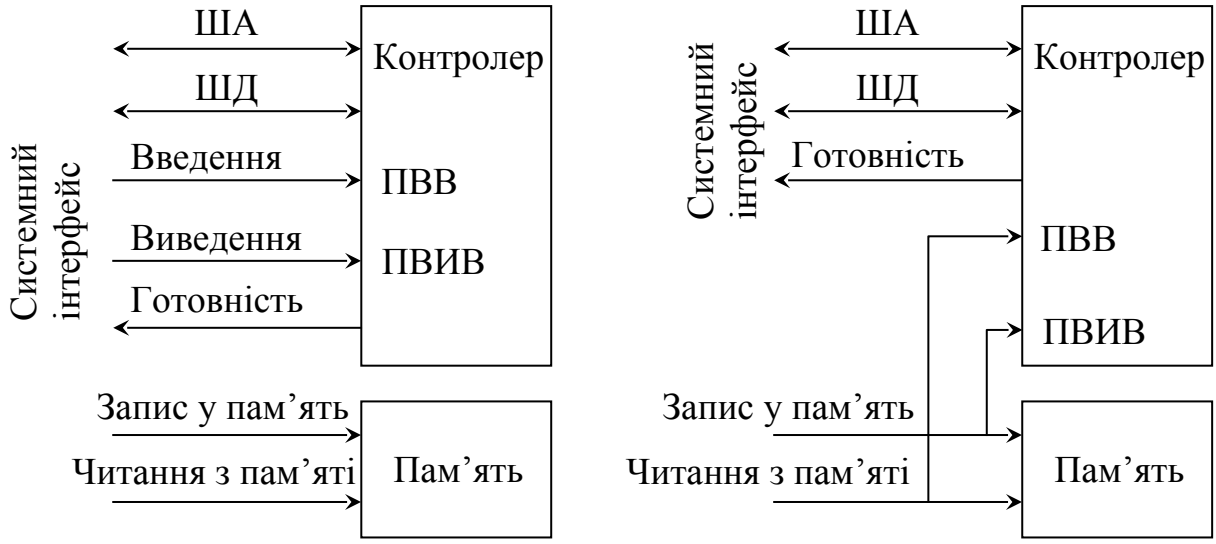
– за аналогією зі зверненням до комірок пам’яті за адресою.

У першому випадку адреса пристрою передається по тій самій шині адреси, що й адреси комірок пам’яті, і трактується як його номер тільки при формуванні спеціальних керувальних сигналів Введення або Виведення з пристроїв введення-виведення, які ініціюються відповідними командами МП. Для синхронізації роботи процесора та контролерів, тобто для указання моментів часу, які визначають готовність даних у пристрої введення для передачі їх у МП або підтверджуючих їх приймання при передаванні з МП у пристрій виведення слугує сигнал Готовність, який надходить з пристроїв введення-виведення.

Для організації програмно-керованого обміну даними з пристроями введення-виведення (ПВВ-ПВИВ) на першому рівні (МП – контролер ПВВ-ПВИВ) достатньо простого набору сигналів (рис. 6.1).

На рис. 6.1,а показано використання керувальних сигналів при виконанні команд введення та виведення. Виконання операції введення починається з того, що МП виставляє на ША адресу ПВВ і сигналом Введення вказує на тип виконуваної операції. За сигналом Введення контролер ПВВ, який адресується, зчитує байт або слово даних з ПП, виставляє на лініях шини даних значення розрядів зчитаного слова та сигналом Готовність сповіщає про це МП. Прийнявши дане через контролер по ШД процесор знімає сигнали з ША та Введення. При виконанні операції виведення МП виставляє на ША адресу

пристрою виведення, на ШД значення розрядів байта або слова, що виводиться, та сигналом Готовність сповіщає процесор, що дані прийняті і можна зняти інформацію з ША та сигнал Виведення.



а) б)

Рисунок 6.1 – Порядок використання керувальних сигналів при введенні-виведенні

При реалізації обміну за аналогією зі зверненням до пам'яті (рис. 6.1,б) використовуються тільки керувальні сигнали Запис у пам'ять та Читання з пам'яті, які використовуються при обміні МП з пам'яттю. Для адрес ПВВ та ПВИВ відводиться визначена частина адресного простору, де можна адресувати необхідну кількість пристроїв.

Спрощена схема організації інтерфейсу з пристроями введення-виведення, яка описана вище, справедлива для багатьох типів МП, на яких побудовані мікропроцесорні системи.

Слід зазначити, що при обміні інформацією МП з периферійним пристроєм слід перевіряти апаратно або програмно готовність його до обміну. Якщо у МПС є кілька ПП, то будують блок формування сигналу Готовність з мультиплексуванням цього сигналу від кількох ПП.

Контрольні запитання:

1 Яку функцію виконують контролери переривань та ПДП у МПС?

2 Які дані треба повідомити контролеру ПДП при його програмуванні?

3 Яку функцію виконують керувальні сигнали МП при взаємодії з іншими підсистемами МПС?

Контрольні запитання підвищеної складності:

1 Керувальний пристрій знаходиться на відстані 15 м від МПС; якого типу інтерфейс Ви рекомендуєте використати для його підключення?

2 Датчик параметрів стану процесу знаходиться на відстані 30 см від МПС; якого типу інтерфейс Ви рекомендуєте використати для його підключення?

6.2 Асинхронний послідовний адаптер RS-232-C

Вхідний контроль:

1 Які цифрові автомати називаються синхронними, а які асинхронними?

2 Які пристрої можуть використовуватись як джерела синхроімпульсів для цифрових автоматів?

При підключенні ПК до мережі, МПС до периферійних пристроїв використовується послідовне передавання даних, яке не створює жорстких обмежень на довжину лінії. В той самий час шина даних ПК та МПС є паралельною. Для перетворення паралельно поданих даних у послідовності у ВІС послідовних інтерфейсів використовуються регістри зсуву. При передаванні даних вони паралельно записуються у регістр зсуву і з кожним тактовим імпульсом зсуваються праворуч, виходячи із регістра в лінію молодшими розрядами вперед. При прийманні дані з лінії вводяться в регістр зсуву праворуч також молодшими бітами вперед і, після заповнення регістра, дані з нього у паралельному поданні передаються в МПС.

Порт послідовного передавання RS-232-C, який називається також **асинхронним адаптером**, або послідовним інтерфейсом, має багатоцільове призначення:

- підключення миші;
- підключення графобудувачів, сканерів, принтерів, диджитайзерів;
- реалізація зв’язку між двома комп’ютерами з використанням спеціального кабелю та оболонки NORTON COMMANDER;
- підключення модемів для передавання даних телефонними лініями;
- підключення до мережі персональних комп’ютерів.

У складі кожного комп’ютера є хоча б один послідовний порт для обміну даними.

Протокол обміну складається з опису передаваного сигналу та призначення його складових частин.

Послідовне передавання даних означає, що дані передаються з використанням однієї лінії. На рис. 6.2 наведено форму електричних сигналів та формат даних при передаванні ASCII коду літери А (41H).

Рівень логічного нуля становить +15 В, логічної одиниці –15 В. Початковий стан лінії – одиничний. Стартовий біт сигналізує про початок передавання даних; далі передаються біти даних у такому порядку: D0–D1–D2–D3–D4–D5… Якщо використовується біт парності Р, то передається й він. Біт парності має таке значення, щоб у пакеті бітів загальна кількість одиниць була парною або непарною залежно від перевірки на парність або непарність. У кінці передаються один чи два стопових біти, які завершують передавання, після чого рівень лінії знов установлюється одиничним до появи наступного

стартового біта. Використання парності, стартових та стопових бітів визначає протокол обміну даними.

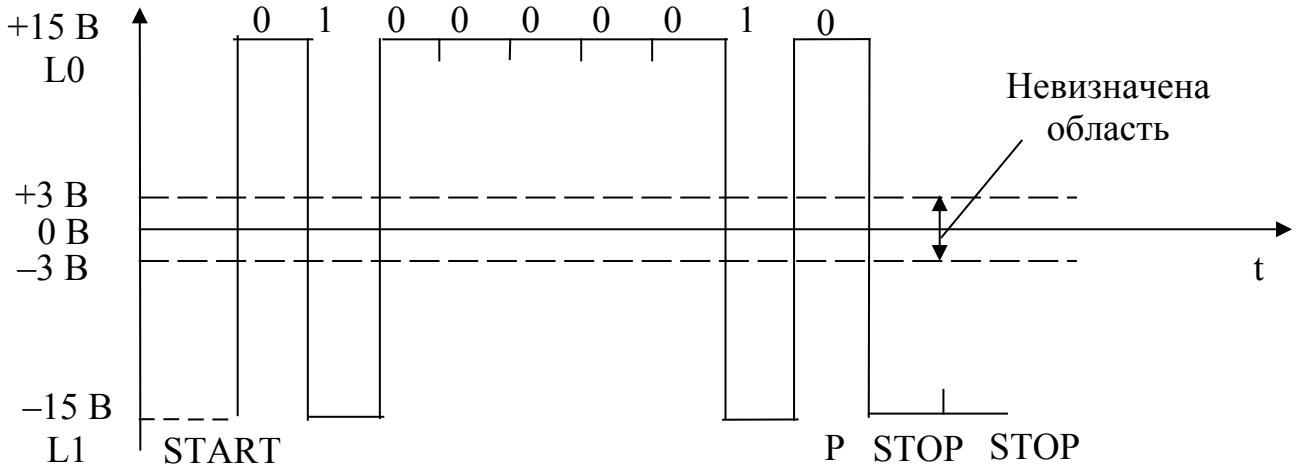


Рисунок 6.2 – Часова діаграма електричних сигналів та формату даних для даного 41H

Іншою важливою характеристикою є швидкість передавання даних. Вона повинна бути однаковою для передавача та приймача. Швидкість передавання даних вимірюється у Бодах (Бод – це кількість бітів, які передаються за секунду). При цьому враховуються і стартстопні біти і біт парності. Іноді використовується інший термін – біти за секунду (bps). Він означає ефективну швидкість передавання даних без урахування службових бітів.

Комп’ютер має від одного до восьми портів послідовного передавання даних, які зреалізовуються на мікросхемі Intel 8250. Це універсальний асинхронний приймач/передавач (UART – Universal Asynchronous Receiver Transmitter). Мікросхема вміщує кілька внутрішніх регістрів, доступних через команди введення-виведення. При передаванні байт записується у буферний регістр передавача, звідки потім переписується у зсувовий регістр передавача. Байт „висувається” з регістра по бітах, молодшими бітами вперед. Аналогічно, приймач теж має зсувовий та буферний регістри. Зовнішні пристрої підключаються до порту введення-виведення через з’єднувач типу DB25P або DB9P, які мають відповідно 25 та 9 виводів. У табл. 6.1 подається призначення контактів з’єднувача.

На етапі ініціалізації комп’ютера модуль POST BIOS тестує наявні асинхронні адаптери та ініціалізує перші два з них. Їхні базові адреси розташовано в області даних BIOS, починаючи з адреси 0000:0400H. Перший адаптер, COM1, має базову адресу 3F8H і займає діапазон адрес з 3F8H до 3FFH. Другий адаптер, COM2, має базову адресу 2F8H і займає адреси 2F8H… 2FFH. Асинхронні адаптери можуть викликати переривання: COM1 – IRQ4 (INT 0CH), COM2 – IRQ3 (INT 0BH).

Таблиця 6.1 – Призначення контактів з’єднувача

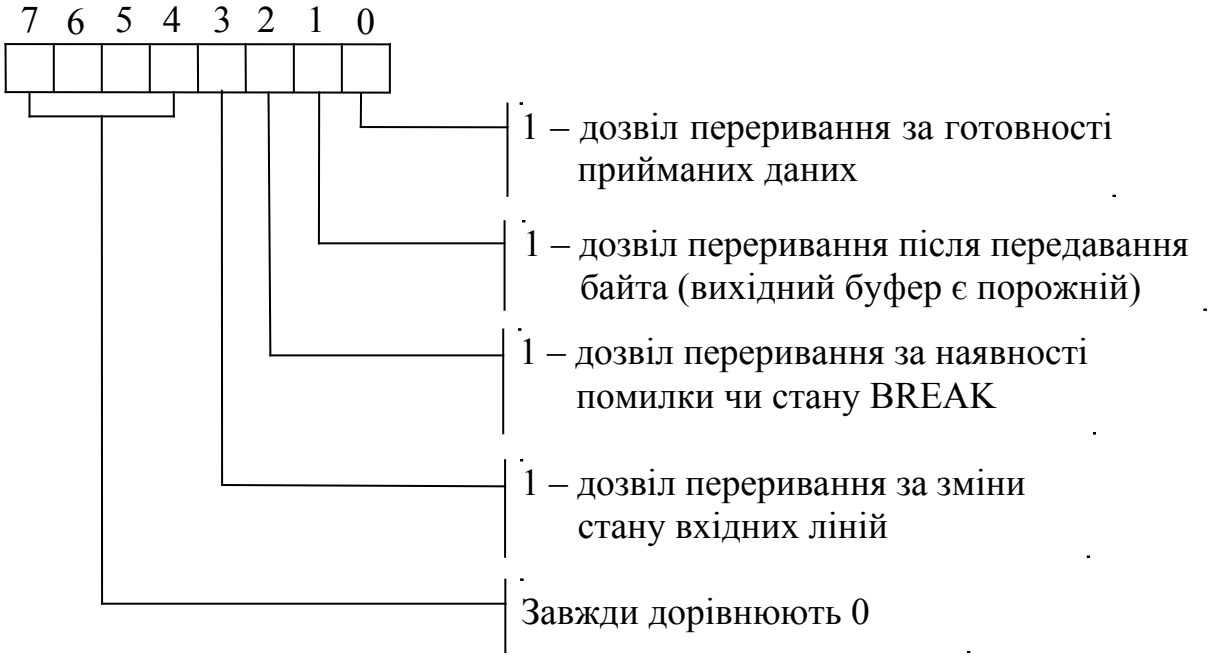
Номер контакту	Призначення контакту	Вхід чи вихід
1	Детектор сигналу з лінії DCD	Вхід
2	Приймані дані	Вхід
3	Передавані дані	Вихід
4	Готовність вихідних даних DTR	Вихід
5	Земля за сигналом	Вихід
6	Готовність даних DSR	Вхід
7	Запит для передавання RTS	Вихід
8	Скидання для передавання CTS	Вхід
9	Індикатор виклику RI	Вхід

Порт 3F8H відповідає регістру даних, які передаються чи приймаються. Для передавання дані треба записати в цей порт. Після прийняття даних від зовнішнього пристрою вони можуть зчитуватись з цього порту.

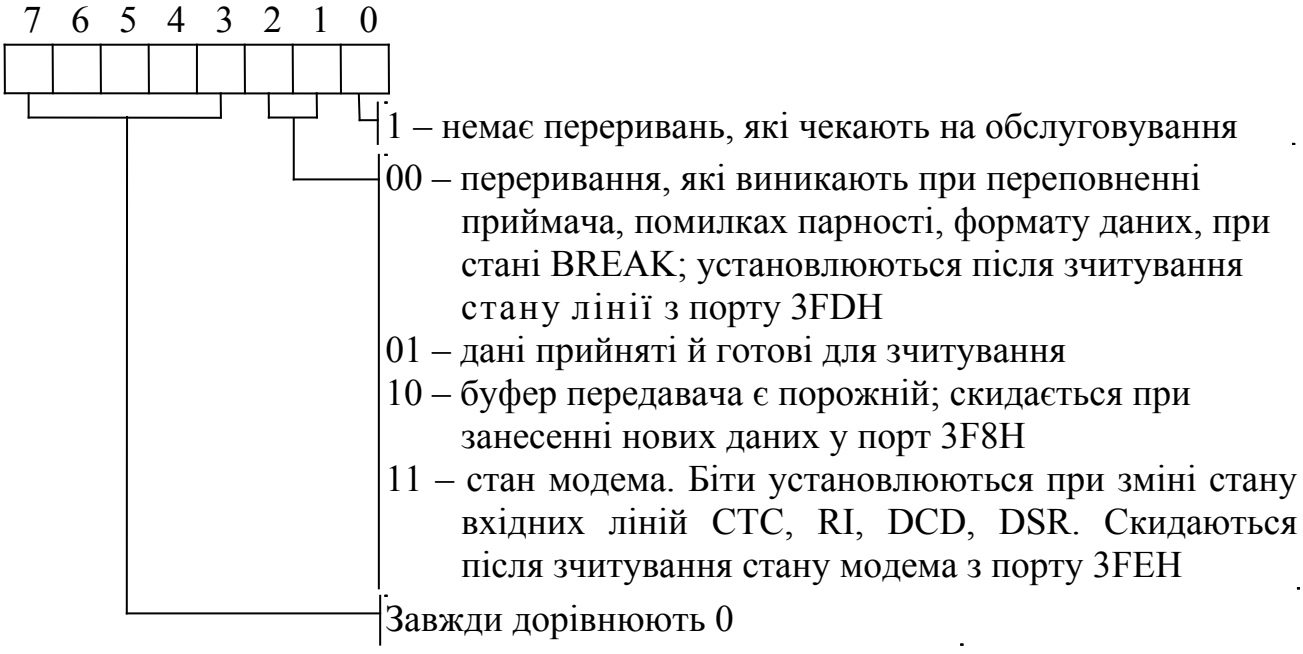
Залежно від стану старшого біта керувального слова, яке виводиться в керувальний регістр з адресою 3FBH, призначення порту 3F8H може змінюватись. Якщо цей біт дорівнює 0, то порт 3F8H використовується для запису передаваних даних. Якщо ж він дорівнює 1, то порт 3F8H використовується для виведення молодшого байта подільника частоти тактового генератора. Змінюючи значення подільника, можна змінювати швидкість передавання даних. Старший байт подільника записується у порт 3F9H. Залежність швидкості передавання даних від значення подільника частоти подано нижче:

Подільник	Швидкість (Бод)
1040	110
768	150
384	300
192	600
96	1200
48	2400
24	4800
12	9600
6	19200
3	38400
2	57600
1	115200

Порт 3F9H використовується як регістр керування перериваннями від асинхронного адаптера або (після виведення в порт 3FBH байта з установленим в 1 старшим бітом) для виведення значення старшого байта подільника частоти. В режимі керування перериваннями порт має такий формат:

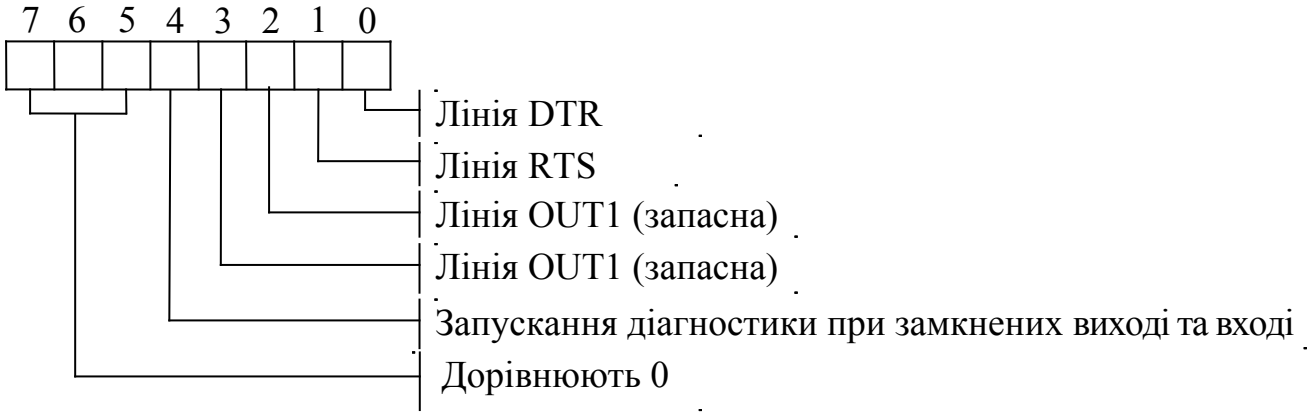


Порт 3FAH призначено для ідентифікації переривань. Його вміст визначає причину переривань. Регістр має такий формат:



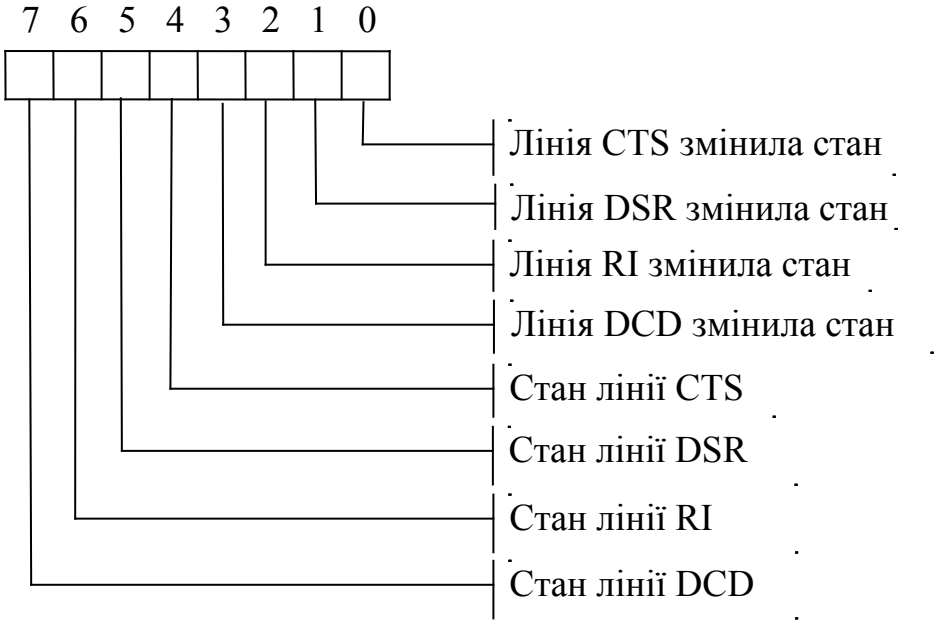
Керувальний регістр 3FBH, доступний для запису та зчитування:

Регістр 3FC керує модемом: станом вихідних ліній DTR, RTS, ліній, специфічних для модемів OUT1 та OUT2, для запускання діагностики при вході асинхронного адаптера, замкненого на його вихід. Формат порту:



Регістр стану лінії 3FDH:

Регістр стану модема 3FEH:



Контрольні запитання:

1 Які функції виконує асинхронний послідовний адаптер?

2 Які пристрої, що входять до складу комп'ютера, підключаються за допомогою адаптера RS-232-C?

3 Який пріоритет мають порти комп'ютера COM1 та COM2?

4 Який режим роботи адаптера задається при завантаженні операційної системи?

5 Задайте керувальне слово, яке забезпечить передавання даного 82H з контролем на непарність та двома стоповими бітами. Наведіть часову діаграму електричних сигналів та формату даних для даного 82H.

6 Яку інформацію вміщує керувальне слово, яке задає режим роботи адаптера?

7 З якою метою зчитується слово стану лінії при прийманні та слово стану буфера при передаванні?

Контрольні запитання підвищеної складності:

1 Назвіть послідовність дій при задаванні бажаної швидкості обміну між RS-232-C та зовнішнім пристроєм?

2 Як заборонити переривання під час обміну даними через RS-232-C?

3 Як перевіряється готовність периферійного пристрою до роботи при обміні даними через RS-232-C?

7 МІКРОПРОЦЕСОРИ

7.1 Архітектура мікропроцесорів

Вхідний контроль:

1 Які мікрооперації може виконувати універсальний регістр?

2 Наведіть приклади пристроїв пам’яті з послідовним доступом та з довільним доступом.

Під архітектурою мікропроцесорів розуміють структурну схему самого МП, програмну (регістрову) модель МП, організацію пам’яті, яку забезпечує МП у складі мікропроцесорної системи, спосіб організації введення-виведення та мову Асемблера, яка керує цим процесором.

З цієї точки зору існують два основні типи архітектури – **фоннеймановська** та **гарвардська**.

Фоннеймановська архітектура показана на рис. 7.1.

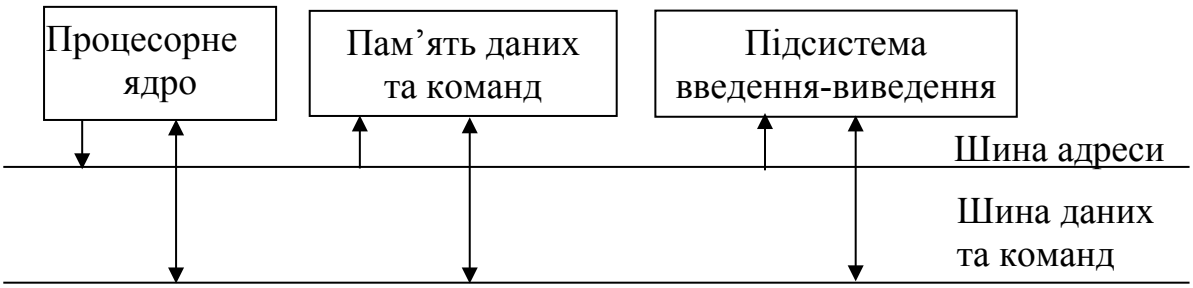


Рисунок 7.1 – Фоннеймановська архітектура

До загальних архітектурних властивостей та принципів побудови фоннеймановської архітектури можна віднести такі особливості:

1 Принцип програми, що зберігається, – це означає, що код програми та її дані знаходяться в єдиному адресному просторі в оперативній пам’яті, доступ до якої здійснюється по одній шині даних та команд.

2 Принцип мікропрограмування – машинна мова, коди, не керують апаратною частиною МП прямо. Кожна команда може бути виконана як результат дії набору сигналів, які треба згенерувати для її фізичної реалізації під керуванням блока мікропрограмного керування. Принцип мікропрограмного керування полягає в тому, що певна комбінація мікрокоманд (зсуву, пересилання інформації, логічних операцій) може створювати набір команд МП.

3 Лінійний простір пам’яті, яку адресує МП, – сукупність комірок пам’яті з послідовним адресуванням.

4 Послідовне виконання команд програми – на послідовних ділянках програми МП вибирає з пам’яті команди строго послідовно. Розгалуження програм виконується за використанням спеціальних команд умовного та безумовного переходів та при зверненні до підпрограм.

5 Дані та команди розміщуються в одному просторі пам’яті у вигляді послідовності нулів та одиниць; процесор не бачить принципової різниці між даними та командами і намагається трактувати вміст деяких послідовних комірок пам’яті як коди машинної команди, а якщо це не так, то програма завершується аварійно. Тому важливо у програмі чітко розподіляти простір даних та команд.

6 Мікропроцесору всеодно, яке логічне навантаження несуть дані, що він їх обробляє.

Класична фоннейманівська архітектура процесора з одним банком пам’яті не дозволяє виконувати багаторазовий доступ до запам’ятовувального пристрою під час виконання одної команди.

Для прискорення оброблення потоків цифрових сигналів у спеціалізованих процесорах цифрового оброблення сигналів використовують так звану гарвардську архітектуру, відповідно до якої процесорне ядро взаємодіє з двома банками пам’яті, як показано на рис. 7.2.

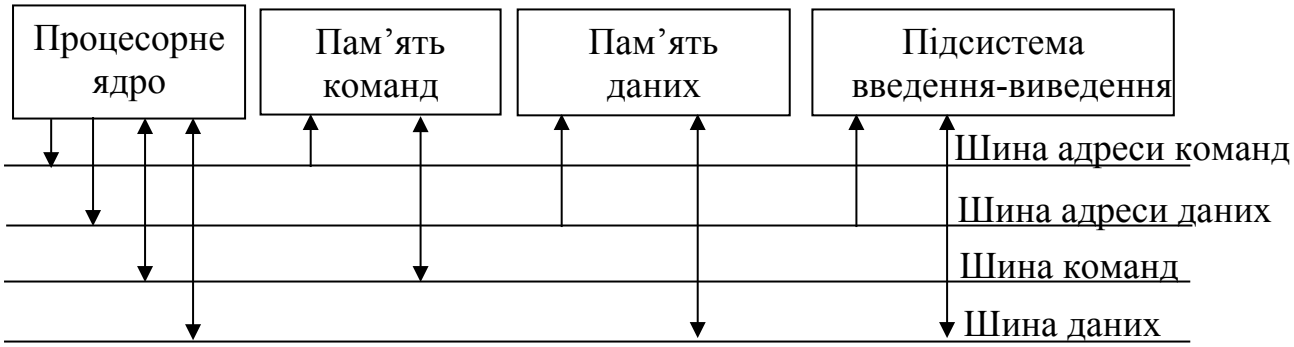


Рисунок 7.2 – Гарвардська архітектура

Звернення до кожного з банків пам’яті виконується за допомогою двох незалежних шин адреси та даних. У гарвардській архітектурі один з банків пам’яті використовується для зберігання програм, а другий для зберігання даних. Частіше використовується модифікована гарвардська архітектура. У цьому випадку один банк зберігає як програми, так і дані, а другий – тільки дані. Гарвардська архітектура дозволяє процесорному ядру за один цикл команди паралельно звертатись до пам’яті даних та пам’яті програм. Деякі мікропроцесори використовують три банки пам’яті з трьома незалежними парами шин адреси та даних (супергарвардська архітектура). Наявність трьох банків дозволяє за один командний цикл виконувати три паралельних звернення до пам’яті: вибирати команду та два операнди.

Контрольні запитання:

1 Які особливості фоннейманівської архітектури мікропроцесорів гальмують підвищення їхньої потужності?

2 При програмуванні фоннейманівських процесорів хто повинен слідкувати за чітким розподілом адресного простору даних та команд?

3 Чи є пряме керування кодами програм апаратною частиною МП?

4 Чи можна передбачити напрямок розгалуження при виконанні команд умовного переходу?

5 Які особливості гарвардської архітектури дозволяють прискорити оброблення цифрових сигналів?

Контрольні запитання підвищеної складності:

1 Що таке модифікована гарвардська архітектура?

2 Що таке супергарвардська архітектура?

7.2 МП фірми Intel

Вхідний контроль:

1 Які особливості має n-МДП-технологія виготовлення ВІС?

2 Чи може працювати МП поза мікропроцесорною системою?

7.2.1 Історична довідка про розвиток мікропроцесорів фірми Intel (Для самостійного вивчення)

Напівпровідникова промисловість почала розвиватися в 50-ті роки XX століття у лабораторіях фірми Bell Telephone Laboratories (Bell Labs) у штаті Каліфорнія, США, в долині Санта Клара, яка сьогодні відома як “Селіконова долина”. Компанія Intel (Integrated Electronics) була заснована Робертом Нойсом та Гордоном Муром у 1968 році, а у 1972 році фірма Intel створила свій перший 8-розрядний мікропроцесор 8008, який мав фоннейманівську архітектуру і багато ознак усього сімейства мікропроцесорів: НОЗП, засоби організації стека, систему переривань і міг слугувати як центральний процесор мікроЕОМ, а також виконувати задачі керування в складі контролерів. Мікропроцесор 8008 був виготовлений за *n*-канальною МОП-технологією, виконував 30000 операцій за секунду та адресував 16 кбайт пам’яті, що було недостатньо для використання його щодо вирішення широкого кола задач. У 1974 році фірма випустила МП І8080, який мав розширену систему команд мови Асемблер і був сумісний з МП 8008. Вперше був виконаний принцип сумісності знизу вверх, тобто користувачі програмного забезпечення МП 8008 могли виконувати його на МП І8080. Швидкість обчислень МП І8080 становила 200000 операцій за секунду, він адресував 64 кбайт пам’яті.

Розробка МП І8080 поставила фірму Intel з річним обсягом понад 1 мільярд доларів у ряд найбільших компаній з виробництва надвеликих інтегральних мікросхем у США.

До 1978 року розвиток МП фірми Intel відбувався шляхом удосконалення та розширення функцій МП І8080.

Аналогом МП І8080 російського виробництва був МП К580ВМ8А, який випускався з метою побудови керувальних МПС, зокрема контролерів жорстких дисків, на ньому також виконувались простіші мікроЕОМ. Частота роботи МП становила 2 МГц.

Структурна схема 8-розрядного мікропроцесора КР 580ВМ80А показана на рис. 7.3.

Для зберігання операндів, які беруть участь в операціях АЛП, передбачено сім 8-розрядних регістрів. Регістр А, який називається **акумулятором**, призначений для обміну інформацією з зовнішніми пристроями; при виконанні арифметичних, логічних операцій та операцій зсувів він є джерелом операнда, в нього пересилається результат виконаної операції.

Шість інших 8-розрядних регістрів B, C, D, E, H, L утворюють блок регістрів загального призначення (РЗП). Ці регістри можуть використовуватися як одиночні або об'єднуватись у регістрові пари BC, DE, HL.

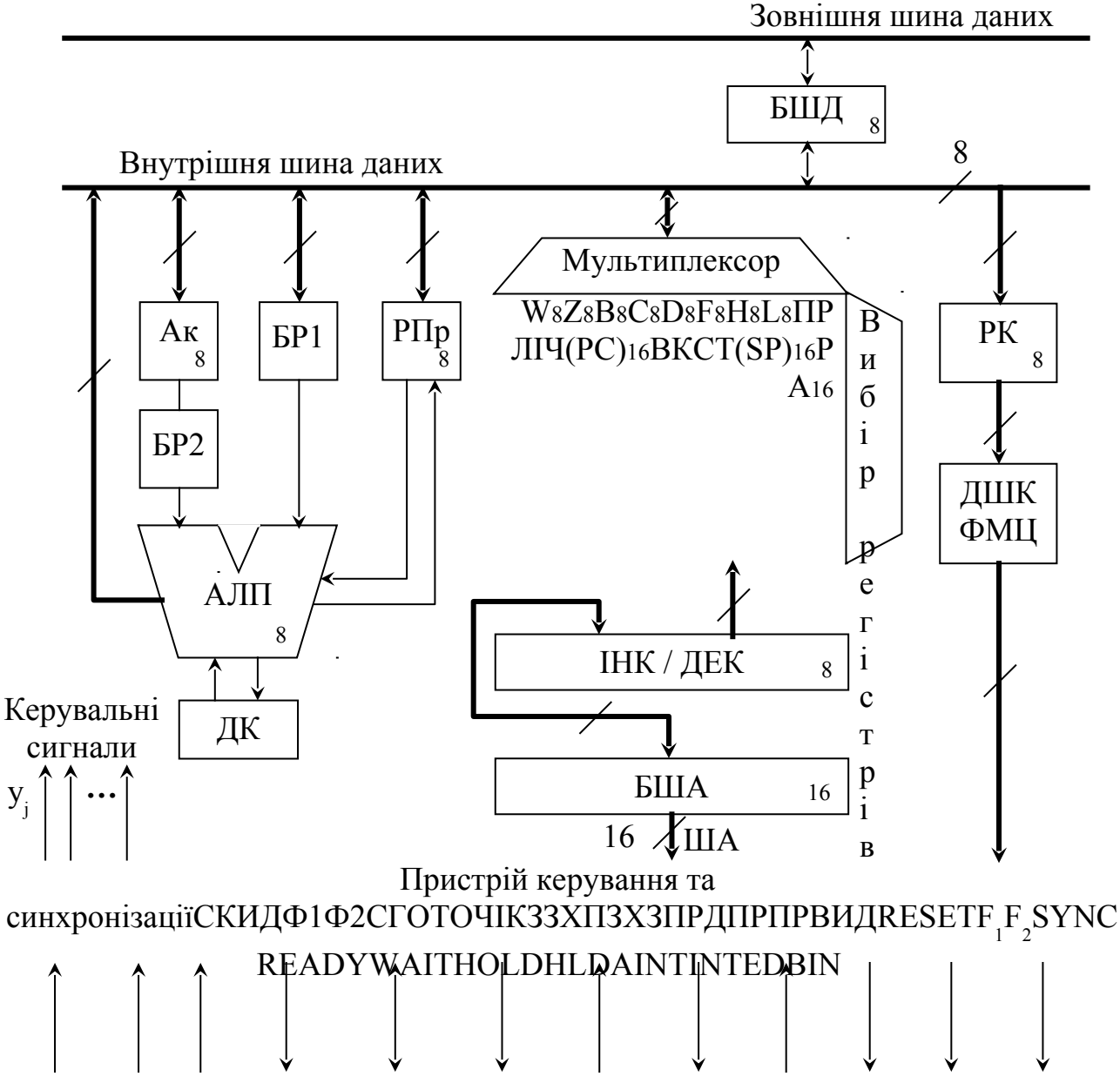


Рисунок 7.3 – Структурна схема 8-розрядного мікропроцесора КР 580ВМ80А

Регістри БР1, БР2, W, Z використовуються як буферні, програмно-недоступні регістри.

Вказівник стека – ВКСТ – (SP) – 16-розрядний регістр, слугує для адресації особливо організованої пам'яті, яка називається стеком.

Програмний лічильник – ПРЛІЧ – (PC) або лічильник команд призначений для зберігання адрес команд; після вибирання з оперативної пам'яті поточної команди вміст лічильника команд збільшується на кількість байтів цієї команди і, таким чином, за відсутності умовних та безумовних переходів у програмах формується адреса наступної команди. Блок регістрів вміщує також спеціальну схему інкрементування/декрементування (ІНК/ДЕК), яка забезпечує без АЛП у процесі передавання даних між регістрами модифікацію вмісту регістрів на 1.

При зверненні до пам'яті з метою вибирання даних, але не команд, як адресу можна використовувати вміст будь-якої регістрової пари з РЗП.

Арифметично-логічний пристрій – АЛП (ALU) виконує арифметичні операції складання з передаванням перенесення у молодший розряд та без урахування цього перенесення, логічні операції кон'юнкції, диз'юнкції, складання за модулем 2, порівняння, а також чотири види циклічних зсувів над 8-розрядними операндами.

АЛП виконано на основі комбінаційної схеми суматора з власними регістрами тимчасового зберігання даних БР1 та БР2.

При виконанні арифметичних та логічних операцій одним з операндів слугує вміст акумулятора і результат операції розміщується в акумуляторі. Циклічний зсув виконується тільки над вмістом акумулятора.

Схема десяткової корекції (ДК) дозволяє виконувати операції над даними, які подані у двійково-десятковій системі числення. При зберіганні операнда, трактованого як десяткове число, розряди кожного регістру, в якому воно вміщене, поділяються на дві групи по чотири розряди кожна і в кожній групі розрядів зберігається одна десяткова цифра, подана у коді BCD (8421). Таким чином, у регістрі можна зберігати 2-розрядне десяткове число. При виконанні арифметичних операцій над двійково-десятковими числами може знадобитися корекція результату (додавання до нього числа 0110_2). Така корекція у кожній 4-розрядній групі результату виконується схемою ДК при виконанні окремої команди мови програмування.

Регістр прапорців РПр, або регістр ознак, складається з п'яти тригерів, які призначені для зберігання певних ознак, якими можна охарактеризувати результат виконання операції АЛП.

Ознака CY (ознака перенесення, від Carry) – перенесення із старшого розряду при виконанні арифметичних операцій та вміст висуваного з акумулятора розряду при виконанні операцій зсувів.

Ознака Z (ознака нуля, від Zero) – установлюється в стан 1, якщо результат операції дорівнює нулю.

Ознака S (ознака знаку, від Sign) установлюється в стан, який визначається значенням старшого розряду результату; S = 1 означає, що результат позитивний, S = 0 – негативний.

Ознака P (ознака парності або непарності кількості одиниць у результаті, від Parity) установлюється в 0, якщо кількість одиниць у результаті непарна, і в 1 – якщо парна; ознака P використовується у багатьох випадках, коли треба перевірити правильність передавання даних за парністю або непарністю кількості одиниць.

Ознака AC (ознака додаткового перенесення від Auxiliary Carry) установлюється в 0, якщо перенесення з молодшої тетради у старшу відсутні, і в 1, якщо має місце; ознака AC використовується при виконанні операцій над двійково-десятковими числами, за наявністю AC за необхідності підключається схема десяткової корекції.

Усі регістри та регістрові пари можуть бути скомутовані за допомогою мультиплексора (MUX) та схеми вибору регістрів як з внутрішньою шиною даних, так і з ША. Регістри акумулятора (A) та регістр прапорців (F) також можуть утворювати регістрову пару AF, яка називається **словом стану програми** (ССП або PSW – PROGRAM STATUS WORD). ССП зазвичай використовується для зберігання в стеку стану програми при зверненні до підпрограм обробки переривань.

При передаванні адреси вміст регістрових пар зберігається у 16-розрядному регістрі адреси (РА), з якого далі через буфер шини адреси (БША) надходить на 16-розрядну шину адреси (ША) і далі в оперативну пам’ять. Число кодових комбінацій 16-розрядної адреси дорівнює 2^{16}, кожна з них може визначати адресу (номер) одної з комірок оперативної пам’яті. Таким чином, забезпечується можливість звернення до пам’яті, яка вміщує до 64 кбайт, тобто 65536 8-розрядних слів.

Буферні регістри даних (БР) та буфер шини адреси (БША) забезпечують зв’язок процесора з зовнішніми шинами даних та адрес. Особливістю буферів є те, що у кожному розряді вони мають логічні пристрої з трьома стабільними станами: крім станів L0 та L1 передбачений третій стан, в якому вони мають безкінечний вихідний опір і є відімкнені від відповідних шин.

Регістр команд (РК) приймає з шини даних код операції команди, який потім декодується у дешифраторі коду (ДШК). Формувач машинних циклів (ФМЦ) організовує взаємодію між усіма блоками і схемами МП та пристроями МПС.

Пристрій керування та синхронізації реалізований апаратно з використанням програмованої логічної матриці. Вихідні сигнали пристрою надходять як до вузлів мікропроцесора, наприклад, вказуючи код операції АЛП, так і для виконання мікрооперацій в МПС.

З шини керування на пристрій керування та синхронізації надходять такі сигнали:

- RESET – скидання або початкового установлення;
- F_1 та F_2 – дві послідовності імпульсів синхронізації з неоднаковими фазами;
- READY – сигнал готовності зовнішнього пристрою до обміну; використовується для організації обміну даними з пристроями, які мають низьку швидкодію порівняно з МП;

− HOLD – сигнал запиту прямого доступу до пам’яті (ПДП) або запиту на захоплення шин; використовується для організації обміну даними з пристроями, швидкодія яких більша ніж у МП;

− INT – сигнал запиту передавання від зовнішнього пристрою, коли він є готовий до роботи.

На шину керування з пристрою керування та синхронізації надходять такі сигнали:

− SYNC – вихід сигналу синхронізації, високий рівень цього сигналу свідчить про те, що по шині даних передається байт стану, який використовується для формування керувальних сигналів для зовнішніх пристроїв;

− WAIT – сигнал підтвердження очікування; його високий рівень свідчить про те, що процесор знаходиться у режимі очікування і виконує такти очікування (холості такти);

− HLDA – сигнал підтвердження прямого доступу до пам’яті або підтвердження захоплення шин; високий рівень цього сигналу свідчить про те, що процесор перевів свої шини адреси, даних та керування у стан високого опору;

− INTE – сигнал дозволу на переривання;

− DBIN – сигнал читання; його високий рівень свідчить про те, що двоспрямована шина даних знаходиться у режимі прийому інформації;

– $\overline{WR}$ – сигнал запису; низький рівень цього сигналу свідчить про те, що двонаправлена шина даних знаходиться у режимі видачі інформації, високий – її прийому.

До групи синхронізуючих сигналів відносяться F_1, F_2**,** SYNC**.** До групи сигналів, які інформують МП про стан зовнішніх пристроїв, можна віднести сигнали RESET та READY. Сигнали запитів на дозвіл роботи від зовнішніх пристроїв та відповідні сигнали від МП – це сигнали HOLD, HLDA, INT, INTE. Сигнали WAIT, DBIN, $\overline{WR}$ свідчать про стан процесора та шини даних.

Програмна модель МП К580ВМ80А

Програмно доступні вузли та елементи архітектури МПС показані на рис. 7.4.

У МП до програмно доступних вузлів відносяться 8-розрядні РЗП B, C, D, E, H, L та A (акумулятор), регістр ознак результату F, 16-розрядний лічильник команд PC (PROGRAM COUNTER), 16-розрядний вказівник стека SP (STACK POINTER), тригер дозволу переривань I (INTERRUPT).

Програмно доступною є пам’ять МПС M (MEMORY): ППЗП, ОЗП та організований на його базі стек.

МПС має ефективну підсистему введення-виведення I–O (INPUT–OUTPUT), яка може складатися з 256 пристроїв введення та 256 пристроїв виведення.

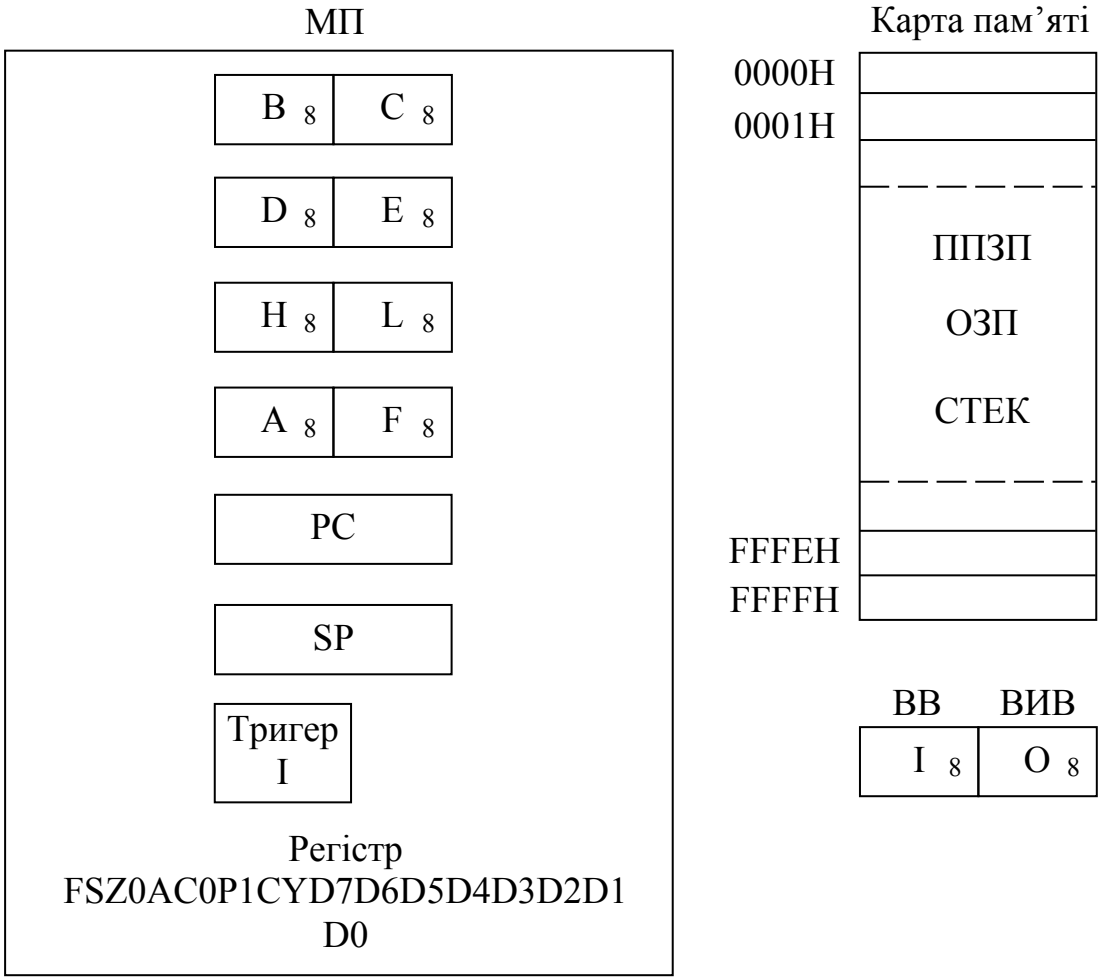


Рисунок 7.4 – Програмно доступні елементи архітектури МПС

Розглянута далі МПС на базі МПК КР580 також є цікавою тільки з точки зору принципів її побудови, взаємодії підсистем, які її складають, та прикладів можливого застосування у телекомунікаціях. Доцільно також на простих прикладах показати виконання окремих команд у МПС.

Мікропроцесорна система КР580

Наведена на рис. 7.5 структурна схема МПС складається з трьох підсистем – центрального процесорного елемента, підсистеми пам'яті та підсистеми введення-виведення і побудована на мікропроцесорному комплекті КР580.

Підсистема центрального процесорного елементу складається з МП типу КР580ВМ80А, генератора тактових імпульсів (ГТІ) типу КР580ГФ24 та системного контролера-формувача (СКФ) типу КР580ВК28. ГТІ виробляє кварцовану двофазову послідовність синхроімпульсів F1 та F2 та СТБС (строб синхронізації в інверсній формі), а також формує вхідні сигнали керування МП – скидання та готовності ГОТ. СКФ формує основні керувальні сигнали ШК.

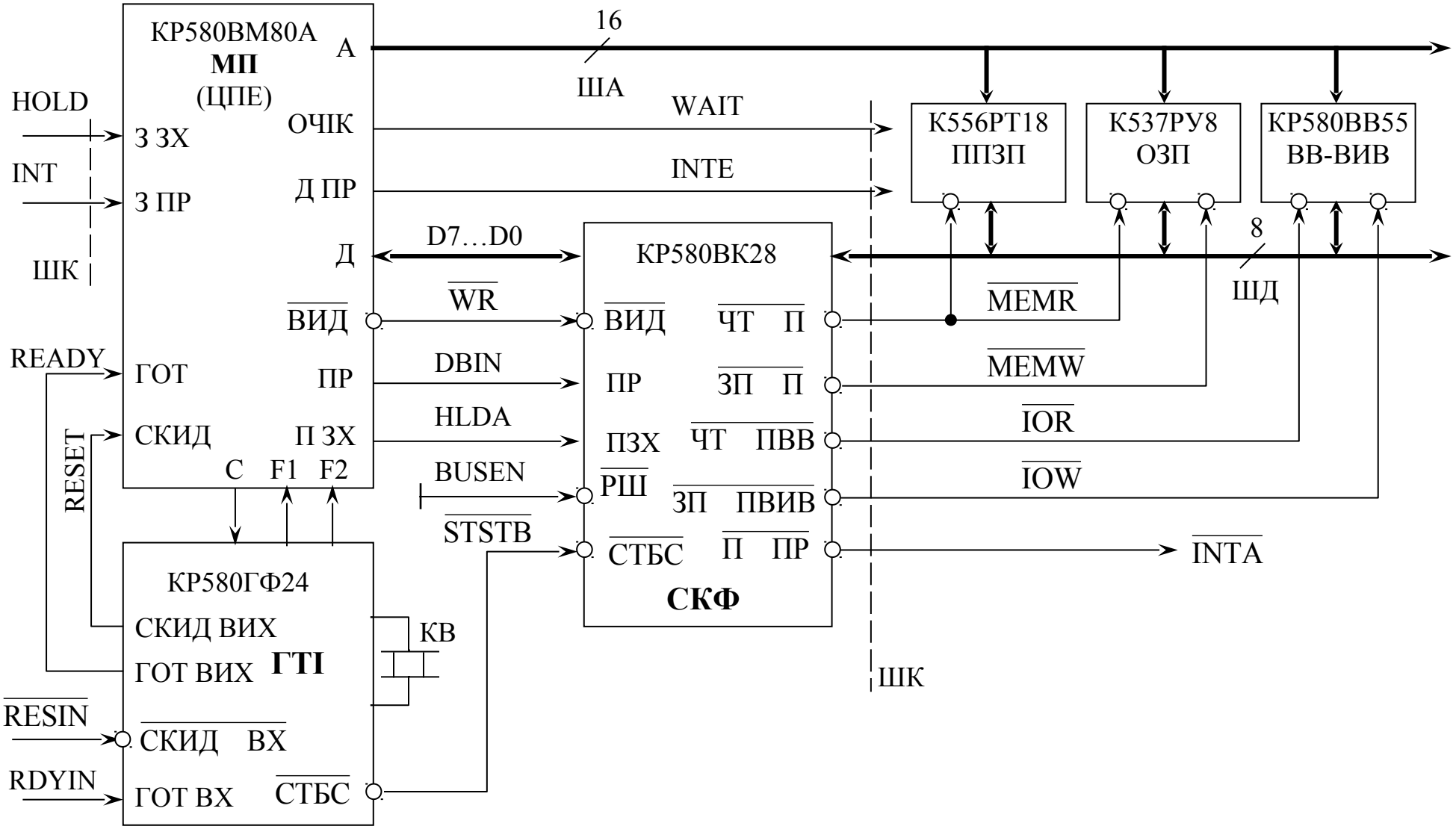


Рисунок 7.5 – Структурна схема МПС КР580

Сигнали видачі даних від МП на ШД для інших підсистем $\overline{\text{ВИД}}$($\overline{\text{WR}}$ - WRITE), прийому даних з ША у МП ПР (DBIN – DATA BUS INPUT) та підтвердження захоплення шин для пристроїв, які працюють у режимі ПДП, ПЗХ (HLDA – HOLD ACKNOWLEDGE) видаються безпосередньо з виводів МП на СКФ, а інші – короткочасно з ШД у вигляді 8-розрядного слова стану процесора ССПЦ (PCSW – PROCESSOR STATUS WORD), яке фіксується у СКФ. ССПЦ має такий порозрядний склад:

D7 – ЧТ П (MEMR – MEMORY READ), читання даних з пам’яті.

D6 – ЧТ ПВВ (INP – INPUT CYCLE), читання даних з пристроїв введення.

D5 – М1, робота МПС у циклі читання та виконання першого байта команди.

D4 – ЗП ПВИВ (OUT – OUTPUT CYCLE), запис/виведення даних через пристрої виведення.

D3 – П ЗУП (HLTA – HALT ACKNOWLEDGE), підтвердження виконання операції зупину МПС.

D2 – СТ (ST – STACK), використання стекової пам’яті.

D1 – ЗП-ВИВ ($\overline{\text{WO}}$ – WRITE-OUTPUT), запис даних у пам’ять або виведення через пристрій виведення.

D0 – П ПР (INTA – INTERRUPT ACKNOWLEDGE), підтвердження переривання МПС.

СКФ синхронізується сигналом $\overline{\text{СТБС}}$ (STSTB – STATE STROBE) – строб синхронізації. За необхідності СКФ можна відключити від ШК установленням його виходів у високоімпедансний стан; для цього на його вхід ДШ (BUSEN – дозвіл шин) треба подати сигнал, який дорівнює 1.

Таким чином на ШК з СКФ подаються такі сигнали:

– $\overline{\text{ЧТ П}}$($\overline{\text{MEMR}}$) – читання даних з пам’яті;

– $\overline{\text{ЗП П}}$ ($\overline{\text{MEMW}}$ – MEMORY WRITE) – запис даних у пам’ять;

– $\overline{\text{ЧТ ПВВ}}$ ($\overline{\text{IOR}}$ – INPUT-OUTPUT READ) – читання даних з пристроїв введення;

– ЗП ПВИВ ($\overline{\text{IOW}}$ – INPUT-OUTPUT WRITE) – запис даних у пристрій виведення;

– $\overline{\text{П ПР}}$ (INTA) – підтвердження переривання;

– $\overline{\text{ОЧІК}}$ (WAIT) – очікування, робота МПС призупиняється;

– $\overline{\text{Д ПР}}$ (INTE – INTERRUPT ENABLE) – дозвіл переривання.

МП може отримувати від різних зовнішніх пристроїв керувальні сигнали:

– З ЗХ (HOLD) – запит на захоплення шин пристроями, які працюють у режимі ПДП;

– З ПР – (INT – INTERRUPT) – запит на переривання роботи МПС;

– $\overline{\text{СКИД ВХ}}$ ($\overline{\text{RESIN}}$ – RESET IN) – СКИД (RESET) – скидання МП, установлення в нуль програмного лічильника РС;

– ГОТ ВХ (RDYIN – READY IN) – ГОТ (READY) – сигнал готовності зовнішніх пристроїв до обміну даними.

СКФ організує буферування ШД та її двонаправленість. Для спрощення на схемі не показані також буфери ША та ШК.

ППЗП призначене для зберігання програм і підпрограм та ОЗП, призначене для тимчасового зберігання проміжних результатів та організації стека, складає підсистему пам’яті. Підсистема введення-виведення може складатися, наприклад, з паралельного або послідовного інтерфейсів, контролерів ПДП або переривань.

Функціонування МПС

Для виконання команди програми необхідна взаємодія якнайменш двох підсистем МПС: підсистеми центрального процесорного елемента та підсистеми пам’яті. Команда зчитується з ППЗП, декодується та виконується. На всі процедури, залежно від типу команди, витрачається від одного до п’яти машинних циклів М1...М5.

У кожному циклі МП один раз звертається до пам’яті або пристрою введення-виведення.

Кожний з машинних циклів складається з трьох-п’яти машинних тактів Т1...Т5, тривалість такту визначається періодом тактових імпульсів ГТІ (рис. 7.6).

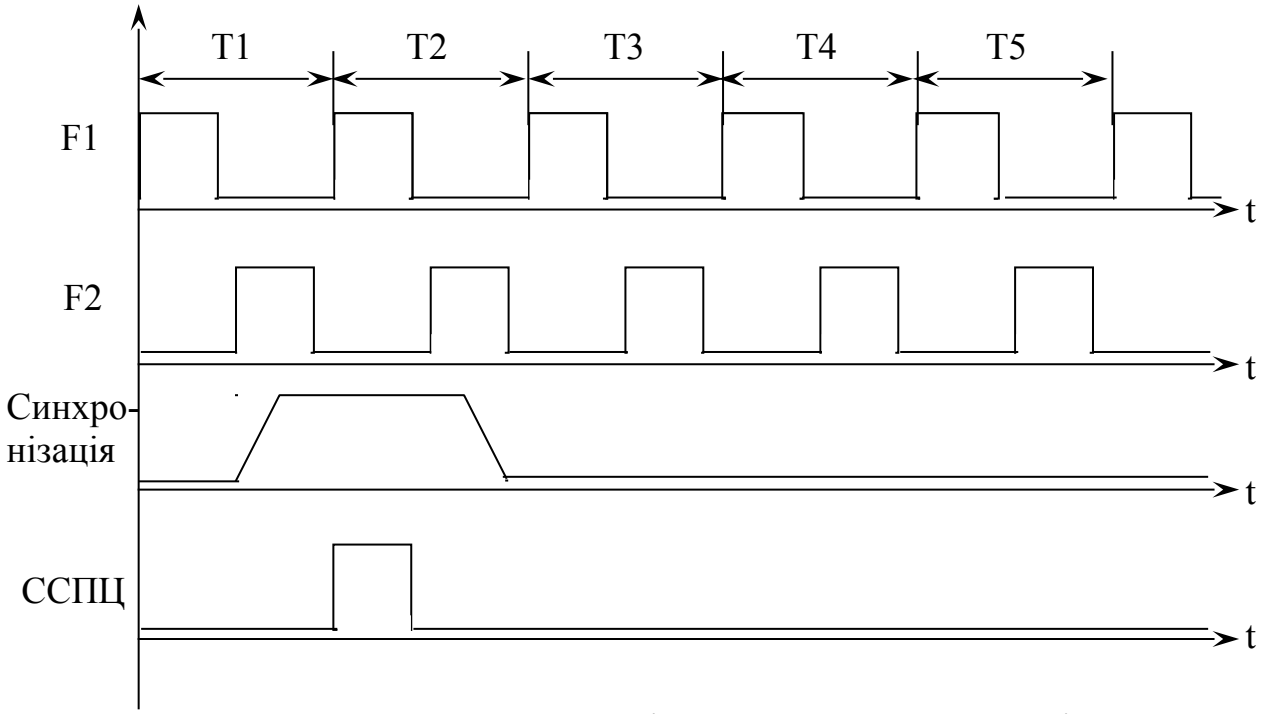


Рисунок 7.6 – Часова діаграма циклу з п'яти тактів

Такти відраховуються від фронтів імпульсів F1. Якщо МП К580ВМ80А працює на частоті 2 МГц, то тривалість одного такту складає 0,5 мкс. За цей час у системі виконується одна або водночас декілька елементарних дій –

мікрооперацій. Набір керувальних сигналів, які керують самим МП, змінюються кожного такту, а в МПС сигнали керування оновлюються тільки на початку кожного машинного циклу команди, тобто за кілька тактів – від 4 до 18 залежно від команди.

У МПС може вироблятися до 10 різних ССПЦ, тобто до 10 різних типів машинних циклів, які показані в табл. 7.1.

Таблиця 7.1 – Типи машинних циклів

№ пп.	Тип циклу	Код ССПЦ
1	Вибірка коду операції команди	01000000
2	Читання пам’яті	01000001
3	Запис у пам’ять	00000000
4	Читання із стека	01100001
5	Запис у стек	00100000
6	Читання з пристрою введення	01000010
7	Запис у пристрій виведення	00001000
8	Дозвіл переривань	11000100
9	Дозвіл останову	01010001
10	Дозвіл переривань під час останову	11010100

Виконання будь-якої команди завжди починається з основного машинного циклу М1, який складається з чотирьох або п’яти тактів:

Т1 – з програмного лічильника (РС) через буфер адреси (БА) на ША МП видає адресу комірки ППЗП з кодом операції виконуваної команди, після чого вміст РС збільшується на 1 (інкрементується); в інших циклах М2...М5 на такті Т1 на ША видається адреса комірки пам’яті для зчитування або запису даних або номер (адреса) пристрою введення-виведення. Одночасно у Т1 кожного циклу в СКФ по ШД надходить слово стану процесора (ССПЦ).

Т2 – ССПЦ фіксується у регістрі СКФ, після чого формуються сигнали ПР (DBIN) та $\overline{\text{ЧТ}}$ $\overline{\text{П}}$ ($\overline{\text{MEMR}}$) для керування подальшим зчитуванням у МП коду операції та його декодування. Одночасно в Т2 аналізуються сигнали готовності ГОТ (READY), запиту захоплення шин ЗЗХ (HOLD) та ЗУП (HALT). При виконанні Т2 у машинних циклах М2...М5 ССПЦ також фіксується в СКФ, але далі виконуються дії залежні від команди.

Т3 – за адресою пам’яті, поданої на ША, зчитується з пам’яті код операції команди та надходить із ШД на регістр коду операції МП (РКОП) для декодування. При виконанні інших циклів М2...М5 у такті Т3 вибираються дані з пам’яті, пристроїв введення, стека або дані виводяться в пам’ять, пристрої виведення та стек.

Т4, Т5 – у МП розшифровується прийнятий код операції команди, визначається її формат, виконуються однобайтові команди або формуються машинні цикли М2...М5 для виконання дво- та трибайтових команд. При виконанні циклів М2...М5 у Т4 виконуються дії залежно від команди, а Т5 може бути відсутній.

На рис. 7.7 показані спрощені часові діаграми циклів вибірки та читання пам’яті, а на рис. 7.8 – часові діаграми циклу запису в пам’ять.

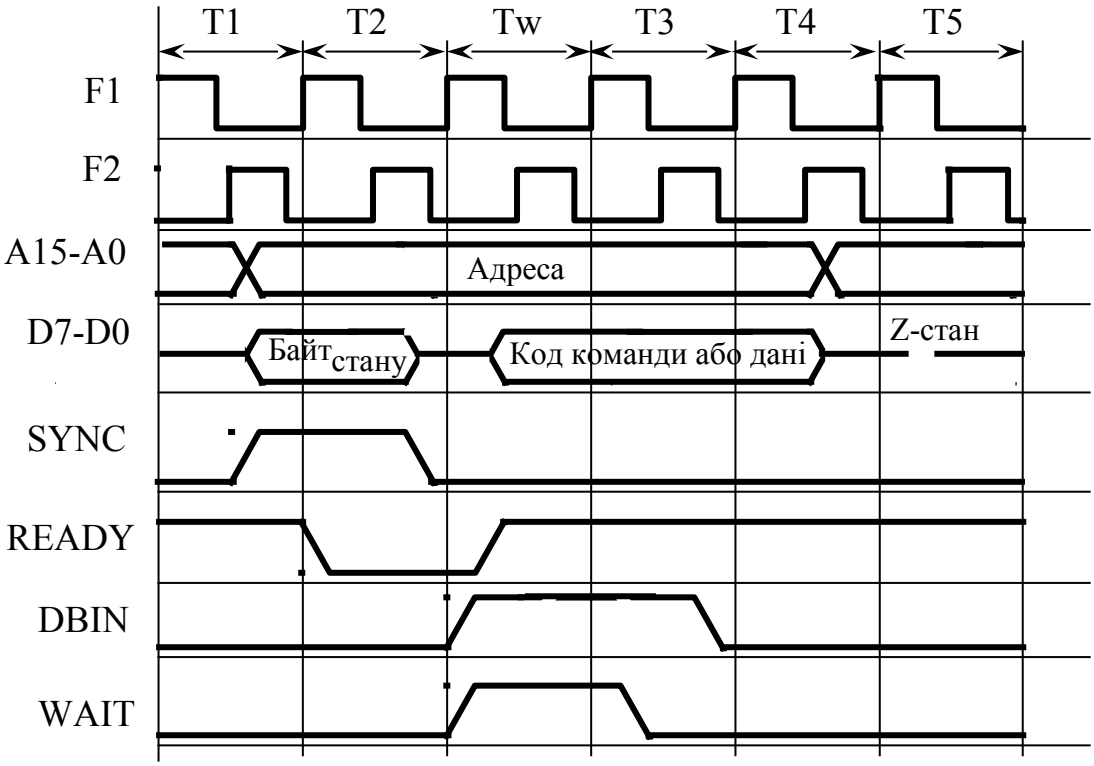


Рисунок 7.7 – Часові діаграми циклів вибірки (читання пам’яті)

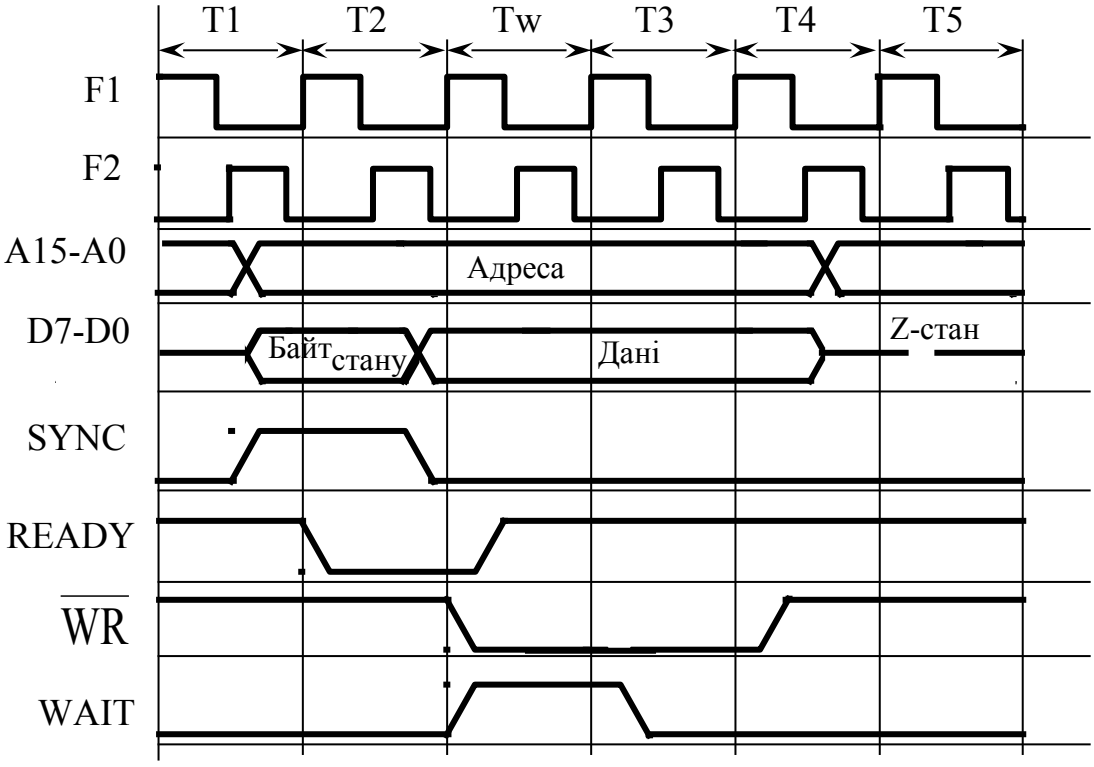


Рисунок 7.8 – Часові діаграмі циклу запису в пам’ять

На діаграмах показано також реакцію МП на наявність сигналу неготовності, який перевіряється в Т2. Якщо зовнішні пристрої неготові до обміну або надійшов сигнал ЗЗХ, сигнал ГОТ = 0 й обмін даними здійснюватись не може і після такту Т2 уставляється один, як показано на рис. 7.7 і 7.8, або кілька тактів очікування Tw залежно від готовності зовнішніх пристроїв або пам'яті.

Робота МПС у режимі переривань

При готовності до роботи асинхронний зовнішній пристрій, підключений до контролера переривань, надсилає у МПС сигнал запиту переривання програми ЗПР (INT). Цей запит може бути задовільнено лише при дозволі переривань, який задається в основній програмі командою EI (ENABLE INTERRUPT) – дозволити переривання. Вона задає у своєму четвертому такті Т4 внутрішній тригер МП дозволу переривань І (INTERRUPT). Команда DI (DESABLE INTERRUPT) забороняє переривання; у цьому разі запит ЗПР ігнорується. За наявності дозволу переривань МП установлює керувальний сигнал дозволу переривань ДПР (INTE). За запитом переривання поточна команда основної програми повністю завершується, після чого починається виконання машинного циклу оброблення переривань МІ:

МІ1

Т1 – на ША через РА установлюється вміст лічильника команд РС – адреси наступної команди основної програми, а на ШД видається ССПЦ дозволу переривань.

Т2 – ССПЦ фіксується в СКФ, після чого на ШК посилається сигнал підтвердження переривань $\overline{\text{П ПР}}$ ($\overline{\text{INTA}}$), який скидає в нуль тригер дозволу переривань І у МП, і знімається сигнал дозволу переривань ДПР (INTE). У результаті блокуються всі подальші запити переривань до установлення тригера І командою EI в основній програмі або підпрограмі обробки переривань. У цьому такті вміст лічильника команд не інкрементується.

Т3 – з шини даних у РКОП МП завантажується спеціальна однобайтова команда RSTV (RESTART), сформована апаратним способом, з кодом 11AAA111, де AAA – будь-яка комбінація одиниць та нулів, забезпечуючи перехід МПС до одної з восьми підпрограм обслуговування переривань, кожна з яких займає по вісім комірок ППЗП.

Т4, Т5 – у цих тактах команда RSTV дешифрується і формуються два наступних тритактних машинних цикли МІ2 та МІ3.

Після декодування команди RSTV визначаються початкові адреси одної з восьми підпрограм, які обслуговують переривання.

У машинних циклах оброблення переривань МІ2 та МІ3 вміст РС для забезпечення можливості повернення до основної програми після завершення підпрограми обслуговування переривань запам'ятовується у стековій пам'яті, після чого у РС завантажується адреса першої команди одної з підпрограм, які обслуговують переривання. Останньою командою підпрограми завжди є

команда повернення RET (RETURN), при виконанні якої у PC повертається зі стека адреса повернення на чергову команду основної програми для її продовження.

На рис. 7.9 та 7.10 показані спрощені часові діаграми циклу переривань відповідно MI1 та MI2 та MI3.

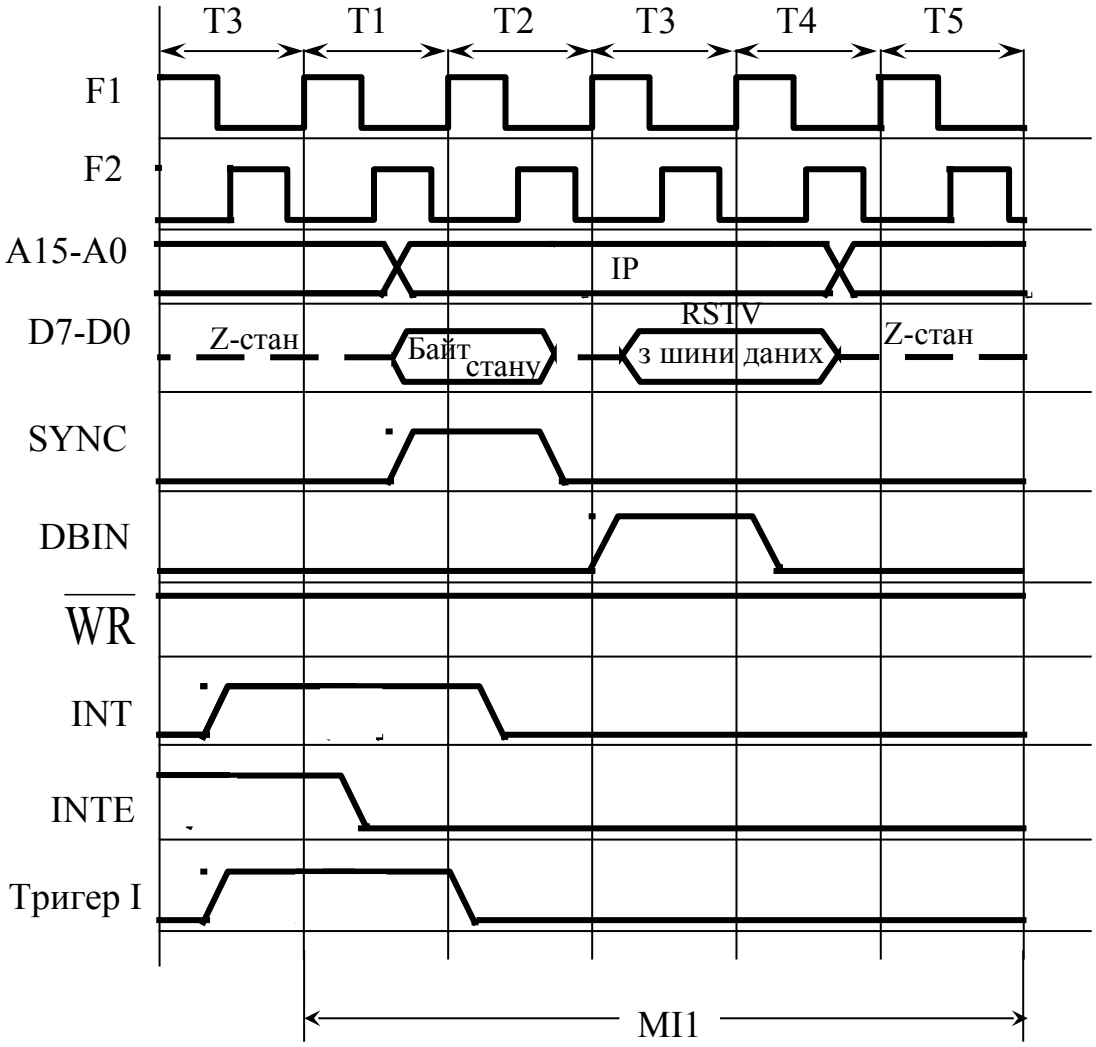


Рисунок 7.9 – Часові діаграми циклу переривань MI1

Можливості режиму переривань можуть бути розширені, якщо замість коду команди RSTV зовнішній пристрій буде формувати код стандартної команди виклику процедур CALL ADDR, де ADDR – початкова адреса підпрограми оброблення переривань.

Розглянуті способи організації переривань називаються **векторними**: ЗП, який запросив переривання вказує початкову адресу першої команди підпрограми оброблення переривань. Є ще багато різних можливостей організації переривань, наприклад невекторні, де задається адреса комірки пам’яті вектора переривань тощо.

Якщо виникають кілька запитів на переривання від декількох зовнішніх пристроїв, то виникають конфлікти, які можна розв’язати у такий спосіб. По-перше, можна послідовно опитувати зовнішні пристрої на наявність від них

запитів на переривання, але це призводить до зниження швидкодії МПС і деякі запити можна перепустити. По-друге, і цей спосіб виявився більш перспективним, можна організувати багарівневі переривання з наданням кожному з пристроїв свого пріоритету. Запити на переривання виконуються за старшинством пріоритетів. При виникненні запиту від пристрою з більш високим пріоритетом обробка переривання з менш високим пріоритетом переривається і може продовжуватись тільки після повного обслуговування більш пріоритетного пристрою. Таких ієрархічних вкладень обслуговуючих підпрограм може бути декілька. Для організації переривань зазвичай використовуються програмовані контролери переривань.

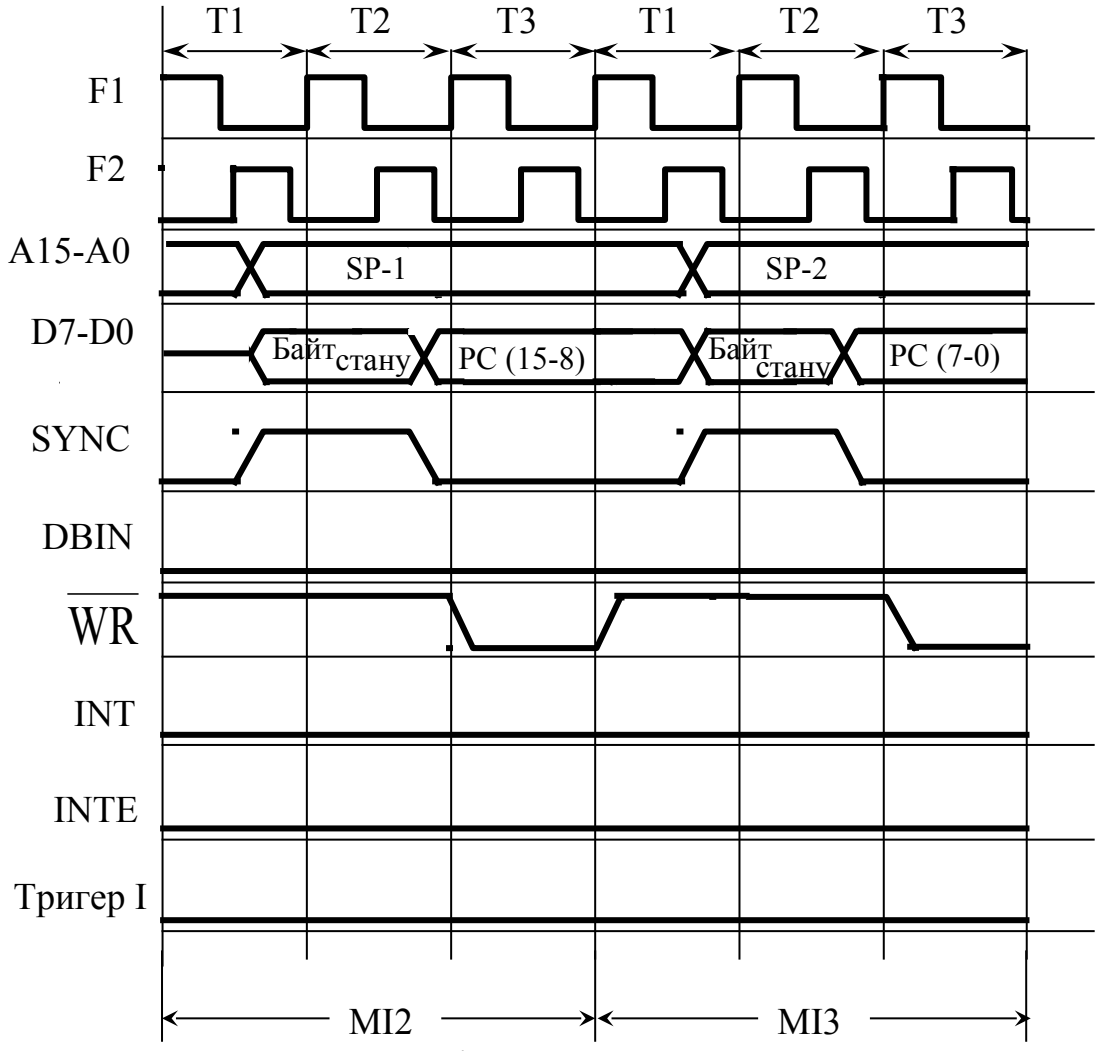


Рисунок 7.10 – Часові діаграми циклу переривань МІ2 та МІ3

Контрольні запитання:

1 В яких засобах обчислювальної техніки використовуються 8-розрядні МП?

2 Яку розрядність має інтерфейс обміну даними з зовнішніми пристроями 8-розрядний МП?

3 Як інтерпретуються дані у 8-розрядних МП фірми Intel?

4 Які є програмно доступні елементи архітектури МПС на МП К580ВМ80А (І8080)?

5 З яких підсистем складається МПС КР580?

6 На прикладі виконання команди пересилання MOV B, C покажіть функціонування МПС.

7 Використовуючи часові діаграми циклів вибірки (читання пам'яті), покажіть реакцію МПС на сигнал неготовності ГОТ = 0.

Контрольні запитання підвищеної складності:

1 Поясніть, як можна за допомогою пріоритетного шифратора підключити до МП зовнішній пристрій з максимальним пріоритетом?

2 В якому режимі працює клавіатура у складі ПК?

3 В якому режимі працює жорсткий диск у складі ПК?

7.2.2 Організація 16-розрядних мікропроцесорів

Вхідний контроль:

1 Як операційна система ПК розміщує коди виконуваної програми в пам'яті?

2 Як у МПС з фоннеймановською архітектурою МП можна розрізняти, чи є код у пам'яті кодом команд, чи кодом даних?

Класичною реалізацією фоннеймановської архітектури є адресування пам'яті за плоскою моделлю, коли вся пам'ять є єдиною лінійною послідовністю байтів. У цій пам'яті зберігаються і дані, і коди програм. Відповідальність за коректне використання пам'яті цілком покладається на прикладного програміста, який повинен забезпечувати цілісність кодів та даних: дані не повинні зіпсувати коди, а стек – перекритися з областю кодів. У багатозадачному режимі, коли кільком десяткам задач надається можливість по черзі виконуватись на віртуальній машині, розподілом ресурсів займається операційна система. Віртуальна (надавана) машина складається з віртуального процесора, віртуальної пам'яті та віртуальної підсистеми введення-виведення. Найбільш незахищеною у багатозадачному режимі є пам'ять. Першим кроком до її захисту було створення у МП І8086, який випускався з 1978 р., моделі пам'яті реального режиму. Ця модель пам'яті організовувала пам'ять у вигляді сегментів коду, стека та даних, які мали різне призначення і різні права доступу. Ця модель була вимушеною і використовувалась з метою можливості адресації обсягу пам'яті 1 Мбайт за допомогою 16-розрядних регістрів, в яких формується адреса комірок пам'яті. Таку модель до цього часу використовують додатки, написані для операційних систем реального режиму типу MS DOS. МП І8086 створювався як призначений для роботи у багатопроцесорних системах з розподіленою пам'яттю та у складі персональних комп'ютерів, тобто вже під керуванням операційної системи.

Для зазначення початку сегмента використовуються спеціальні сегментні регістри: у 16-розрядних МП – CS, DS, SS,ES.

Переміщення програм та даних означає, що вони можуть бути у різні моменти часу в різних областях пам’яті, не вимагаючи від операційної системи або додатків своєї модифікації.

Для визначення фізичної адреси адресованої комірки пам’яті у межах 2^{20} адресного простору вміст сегментних регістрів помножується на 4 (зсувається вліво на 4 двійкових розряди) і до нього додається значення зміщення адреси комірки від початку сегмента.

Сегментування пам’яті визначило архітектуру і програмні моделі мікропроцесорів фірми Intel.

Структурна схема МП I8086 наведена на рис. 7.11.

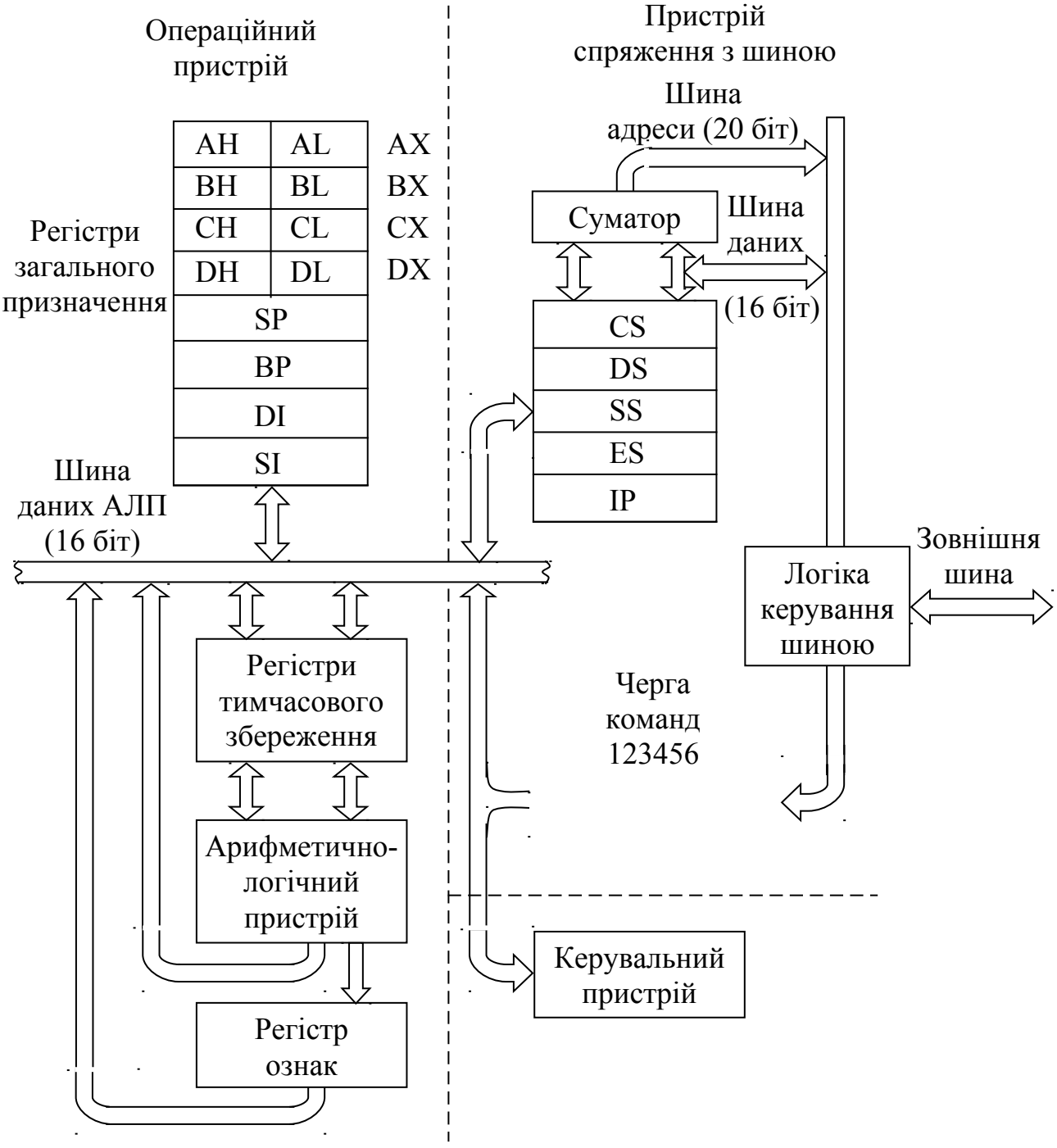


Рисунок 7.11 – Структурна схема МП I8086

За функціональним призначенням вузли схеми можна розбити на операційний пристрій, пристрій спряження з шиною та керувальний пристрій. В операційному пристрої виконуються обчислення, пов'язані з виконанням команд з оброблення даних; у пристрої з'єднання з шиною – обчислення, які забезпечують формування адрес команд та операндів, виклик команд та їх зберігання до початку виконання; керувальний пристрій, який дешифрує коди команд та генерує керувальні сигнали та сигнали синхронізації для МП та МПС. До операційного пристрою входять арифметично-логічний пристрій з регістрами тимчасового збереження і регістром ознак, блока регістрів загального призначення, який складається з чотирьох 16-розрядних регістрів загального призначення AX, BX, CX, DX, кожний з них може використовуватись як два 8-розрядних, регістрів-вказівників SP, BP та індексних регістрів SI, DI, які використовуються при формуванні адрес.

Пристрій з'єднання з шиною складається з шестибайтової регістрової пам'яті, яка називається **чергою команд**, сегментних регістрів CS, DS, SS, ES та вказівника команд IP (Instruction Pointer), суматора та логіки керування шиною.

Черга команд працює за принципом FIFO (First Input – First Output, перший увійшов – перший вийшов). У черзі команд можуть накопичуватись послідовні байти команд обсягом до 6 байт. Це дозволяє подавати команди з випередженням з черги в операційний пристрій, де вони виконуються, і дає можливість сумістити за часом виконання процеси, пов'язані з вибиранням команд із пам'яті, і процеси, пов'язані з їх виконанням. Черга команд називається також **конвеєром команд**.

Швидкодія процесора збільшується за рахунок конвеєризації на лінійних ділянках програм, але коли у чергу попадають команди переходів, викликів підпрограм, у тому числі підпрограм оброблення переривань та повернення з них, черга скидається і вибирання починається з нової адреси програмної пам'яті.

Суматор бере участь у формуванні виконавчої адреси (Executive Address) операндів, які розміщуються у пам'яті, та фізичної адреси операндів і команд.

Процесор має мультиплексовану двоспрямовану 16-розрядну шину даних/адреси AD15–AD0, яка використовується у мінімальному, однопроцесорному режимі. У максимальному режимі при роботі у складі багатопроцесорних систем МП може адресувати 1 Мбайт пам'яті за рахунок додаткових чотирьох ліній адреси/стану A19/S6…A16/S3.

Мікропроцесор I8086 (російський аналог К1810ВМ86) є розвитком МП I8080, їхня архітектура подібна. Програмно-доступні вузли і система команд I8080 є підмножиною вузлів і системи команд МП I8086. Хоча програми, написані машинною мовою I8080, не можуть виконуватись безпосередньо на МП I8086, але можна говорити про сумісність МП фірми Intel знизу уверх; програми достатньо просто переводяться з мови Асемблера I8080 на мову Асемблера I8086 за допомогою їхньої повторної трансляції.

МП I80186 є наступною моделлю МП з архітектурою I8086 і має додатково логічний блок вибирання мікросхеми, два незалежних швидкісних канали ПДП, три програмованих таймери, програмований контролер переривань та вбудований генератор тактових імпульсів. Система команд порівняно з МП I8086 є розширеною і вміщує всі команди попередніх моделей, починаючи з I8080.

У 1983 р. фірма Intel випустила МП I80286, який став основою для розробки ПК IBM PC/AT. МП I80286 може працювати у двох режимах: реальному та віртуальному. У реальному режимі МП I80286 функціонує як МП I8086 з підвищеною швидкодією. У віртуальному режимі фізична адреса комірки пам’яті знаходиться за допомогою таблиць, які індексуються за допомогою сегментних регістрів, тобто реалізується так званий захищений режим роботи МП, який розглядається у підрозділі 7.2.5. При включенні у МП I80286 установлюється реальний режим, ініціалізуються різні регістри і прапорці і виконується команда завантаження слова стану машини, в якому установлено біт дозволу захисту. Після переходу у віртуальний режим повернути його до реального режиму можна тільки апаратним сигналом скидання.

МП I80286 забезпечує засоби захисту різних областей пам’яті при роботі у віртуальному режимі за рахунок наявності рівнів привілеїв виконуваних сегментів кодів. Якщо рівень виконуваної програми є менший за рівень, який надано сегменту, МП вимикає засоби захисту.

МП I80826 адресує до 2 Мбайт реальної пам’яті та 4 Гбайт віртуальної. Система команд вміщує всі команди більш ранніх моделей та кілька нових команд, пов’язаних з реалізацією реального та віртуального режимів.

МП I80826 припускає спряження з співпроцесором I80287, призначеним для виконання операцій з плаваючою точкою. Обидві мікросхеми можуть виконувати команди одночасно.

МП I80826 підтримує багатозадачний режим шляхом перемикання з однієї виконуваної задачі на іншу. МП створює для кожної задачі сегмент пам’яті, який вміщує всю інформацію, небхідну для запуску та зупину задачі (сегмент стану задачі). Основне його призначення – збереження вмісту регістрів задачі на той час, коли задача не виконується.

Контрольні запитання:

1 З якою метою у 16-розрядних процесорах фірми Intel реалізується сегментування пам’яті?

2 Яку роль у структурній схемі МП I8086 (К1810) відіграє пристрій спряження з шиною?

3 Які операції виконує АЛП у складі МП I8086?

4 Які прапорці виставляє АЛП I8086?

Контрольні запитання підвищеної складності:

1 З якою метою МП I8086 має мультиплексовану двоспрямовану шину даних/адреси?

2 Як, на Ваш погляд, можна зберегти цілісність даних, не сегментуючи пам’ять?

7.2.3 Програмна модель МП І8086

Вхідний контроль:

1 Які вузли МП входять до програмної моделі МП?

2 З якою метою кожний регістр МП І8086 має своє функціональне призначення?

Програмна модель МП І8086 показана на рис. 7.12.

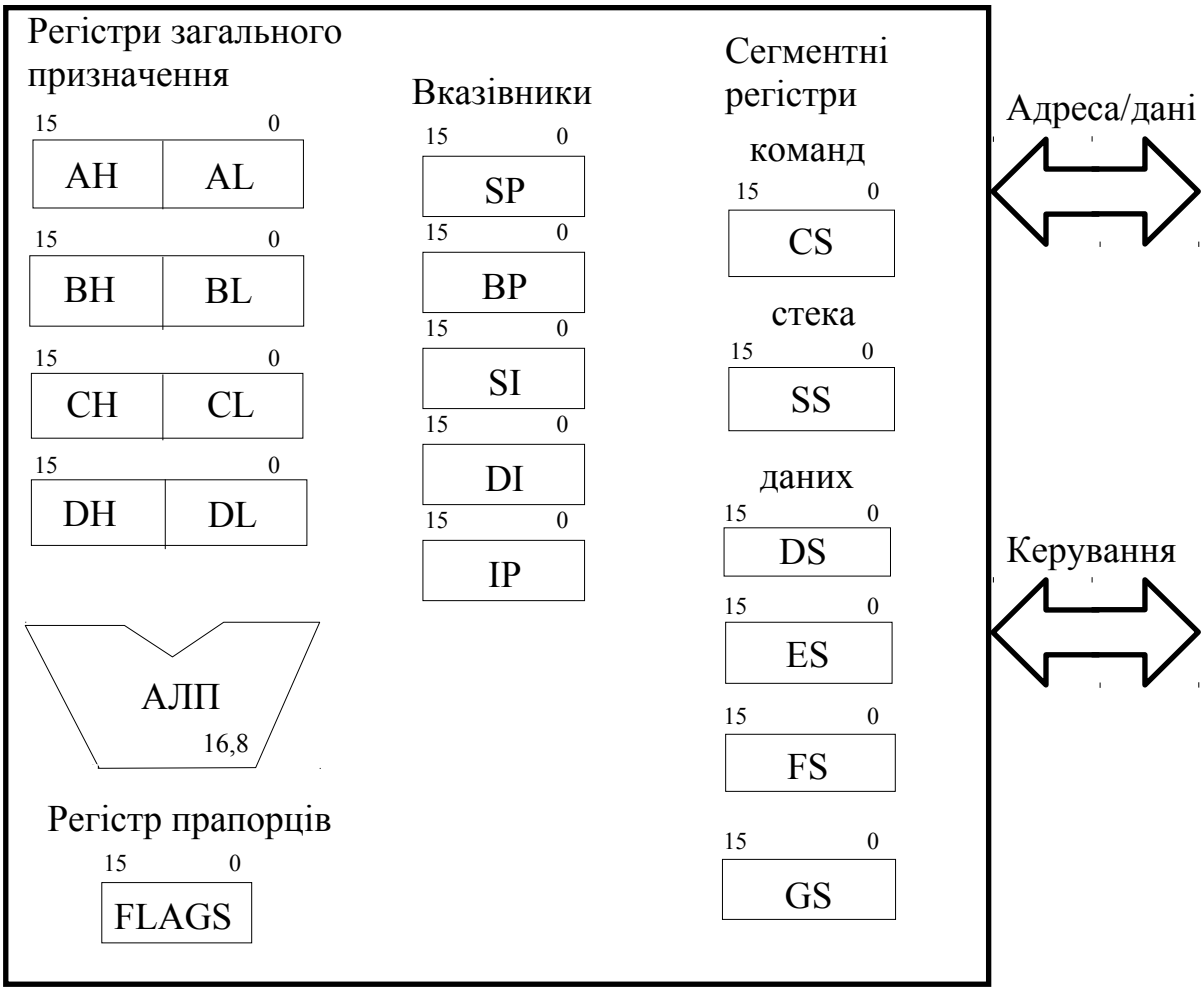


Рисунок 7.12 – Програмна модель МП І8086

Регістри загального призначення РЗП AX, BX, CX, DX допускають окреме використання їх молодших байтів AL, BL, CL, DL і старших байтів AH, BH, CH, DH. Тим самим забезпечується можливість операцій з байтами та словами і створюються умови для програмної сумісності МП І8086 та МП І8080.

Решта регістрів є неділимою, вони оперують 16-бітовими словами навіть у разі використання тільки їх молодшого або старшого байтів. Регістри-

вказівники SP і BP та індексні регістри SI та DI зберігають зміщення адреси у межах поточного сегмента пам’яті.

Регістри загального призначення (РЗП), або регістри даних, призначені для тимчасового зберігання змінних або проміжних результатів:

– AX – регістр-акумулятор – є регістр, в якому зберігається один із операндів перед операцією, що виконується в АЛП, та куди передається результат операції;

– BX – базовий регістр, який може використовуватися для формування базової адреси будь-яких змінних у пам’яті;

– DX – регістр додаткових даних у деяких операціях, в яких результат перевищує довжину розрядної сітки процесора; крім того, у командах введення-виведення цей регістр вміщує адресу порту;

– CX – регістр-лічильник; у командах організації циклів за умовчанням використовується як лічильник циклів.

Група індексних регістрів та вказівників, яку утворюють регістри:

– BP – вказівник бази;

– SI – регістр індексу джерела;

– DI – регістр індексу приймача;

– SP – вказівник стека;

– IP – вказівник команд.

Регістри BP, SI, DI використовуються при формуванні адрес змінних або для зберігання проміжних даних або результатів.

Індексні регістри SI і DI у режимі автоінкрементування та автодекрементування використовуються, здебільшого, для адресації елементів масивів. Вони можуть використовуватись також для зберігання проміжних результатів.

Вказівник стека SP адресує вершину стека, особливо організованої області (сегмента) пам‘яті.

Регістри сегментів CS, SS, DS, ES використовуються для збереження інформації про початкові адреси сегментів пам‘яті:

– CS – регістр сегмента команд;

– SS – регістр сегмента стека;

– DS – регістр сегмента даних;

– ES – регістр додаткового сегмента даних.

Регістр CS разом з регістром вказівника команд IP використовується для обчислення адреси наступної виконуваної команди.

Регістр SS разом з вказівником стека SP використовується для визначення адрес у сегменті стека. При роботі з підпрограмами адреси стекової пам’яті формуються з використанням регістра SS та вказівника бази BP.

Регістр DS визначає область пам’яті, де зберігаються змінні, що використовуються у програмі. Адреси цих змінних визначаються з використанням регістрів BX, SI, DI.

Регістр ES адресує сегмент пам’яті під час операцій з рядками.

Довжина сегментних регістрів становить 16 розрядів, тому операції з сегментними регістрами є операціями зі словами, які називаються **селекторами**.

Регістр вказівника (лічильника) команд IP, призначений для адресації в середині поточного сегмента коду. Вказівник команди прямо у командах не вказується, але бере участь в усіх командах передавання управління (умовних та безумовних переходів, виклику підпрограм, повернення з підпрограм тощо). У програміста немає можливості безпосередньо змінювати вміст регістру IP.

Регістр прапорців FLAGS зберігає інформацію про ознаки результату.

АЛП призначений для виконання арифметичних та логічних операцій над 16-розрядними або 8-розрядними числами.

Регістр прапорців FLAGS вміщує 16 бітів, але не всі з них зайняті ознаками результату. АЛП виставляє 9 прапорців:

– CF(0) – прапорець перенесення, дорівнює 1, коли результат операції виходить за межі розрядної сітки;

– PF(2) – прапорець парності кількості одиничних бітів у молодшому байті результату, установлюється в 1, коли кількість одиничних бітів парна;

– AF(4) – прапорець додаткового перенесення, установлюється в 1, коли є перенесення або позика для третього біта результату;

– ZF(6) – прапорець нульового результату, установлюється в 1, коли результат операції дорівнює 0;

– SF(7) – прапорець знака, дублює стан найстаршого біта результату, при роботі з числами зі знаками SF визначає знак числа – для додатних чисел SF=0, для від’ємних SF = 1;

– TF(8) – прапорець трасування, що використовується при налагодженні програм у покроковому режимі, в який переводиться МП при TF = 1;

– IF(9) – прапорець переривань, якщо IF = 1, переривання дозволені;

– DF(10) – прапорець напрямку, використовується у командах роботи з рядками, якщо DF = 0, вміст регістрів SI та DI збільшується і рядок обробляється зліва направо, при DF = 1 – навпаки;

– OF(11) – прапорець переповнення, OF = 1 установлюється при перебільшенні результату операції над числами зі знаком допустимого діапазону.

Первісне завантаження регістрів загального призначення AX, BX, CX, DX та їхніх частин, а також вказівників та індексних регістрів SP, BP, SI, DI можливе за допомогою безпосередньої адресації. Завантаження сегментних регістрів SS, DS, ES може реалізуватися через акумулятор AX. Регістр командного сегмента CS є програмно недоступний і завантажується системою програмування, наприклад, TASM.

Приклади завантаження регістрів:

```
MOV AX,1234H   ;   Завантаження 16-розрядного регістра AX числом
               ;   1234H
MOV AX,1234H   ;   Завантаження 16-розрядного акумулятора числом
               ;   1234H
MOV AH,34H     ;   Завантаження старших восьми розрядів
```

```
                ;   акумулятора числом 34H
MOV AL,12H      ;   Завантаження молодших восьми розрядів
                ;   акумулятора числом 12H
MOV BP,400H     ;   Завантаження вказівника бази числом 400H
MOV SP,200H     ;   Завантаження вказівника стека зміщенням 200H
MOV SS,AX       ;   Завантаження сегментного регістра SS початковою
                ;   адресою сегмента 3412H з AX
MOV DS,AX       ;   Завантаження сегментного регістра DS початковою
                ;   адресою сегмента 3412H з AX
MOV ES,AX       ;   Завантаження сегментного регістра ES початковою
                ;   адресою сегмента 3412H
MOV SI,400H     ;   Завантаження індексного регістра SI початковим
                ;   зміщенням 400H
PUSH BX         ;   Завантаження вершини стека з BX
POP CX          ;   Перевантаження стека до CX
MOV AX,CS       ;   Запам’ятовування сегментного регістра кодів у AX
PUSH AX         ;   Завантаження AX до стека
POP BX          ;   Перевантаження стека до BX
MOV SS,BX       ;   Завантаження SS з BX
MOV AX,0000H    ;   Обнулення регістра AX
PUSH DS         ;   Ініціалізація стека початковою адресою
PUSH AX         ;   Ініціалізація стека нульовою адресою
```

Контрольні запитання:

1 Яке функціональне навантаження несуть регістри загального призначення у МП I8086?

2 Як обчислюється виконавча адреса у МП I8086?

3 Як реалізується первинне завантаження сегментних регістрів?

Контрольні запитання підвищеної складності:

1 Напишіть фрагмент програми, який би обмінював вміст регістрів BX та DX за допомогою стека.

2 Як можна завантажити, на Ваш погляд, сегментний регістр кодів CS?

7.2.4 Режим переривань МП I8086

Вхідний контроль:

1 Які режими роботи МП Ви знаєте?

2 З якого типу периферійними пристроями повинен працювати МП у режимі переривань?

Переривання – це режим мікропроцесора, коли він тимчасово перериває виконання поточної програми і переходить до підпрограми обробки переривань, яка є більш терміновою або важливою. Після повернення з підпрограми процесор продовжує виконувати поточну програму. Для цього у

стеку запам’ятовується адреса повернення (CS та IP), а також вміст регістра прапорців FLAGS, які потрібні для виконання також підпрограми обробки переривань. Значення IP перед завантаженням у стек корегується до адреси команди, перед якою МП почав обслуговувати переривання. Це пов’язано з наявністю у процесорі конвеєра команд, завдяки якому IP адресує команди з випередженням. Слід зазначити, що вміст регістрів CS, IP та FLAGS запам’ятовується у стеку автоматично, а вміст усіх інших регістрів, які будуть задіяні потім у підпрограмі, треба запам’ятати на її початку, а потім наприкінці її вивантажити зі стеку.

Переривання за видами можуть ідентифікуватись як зовнішні і внутрішні, апаратні та програмні, масковані (ті, що можна забороняти) і немасковані. Всього МП підтримує 256 типів переривань. Можливі джерела переривань показані на рис. 7.13.

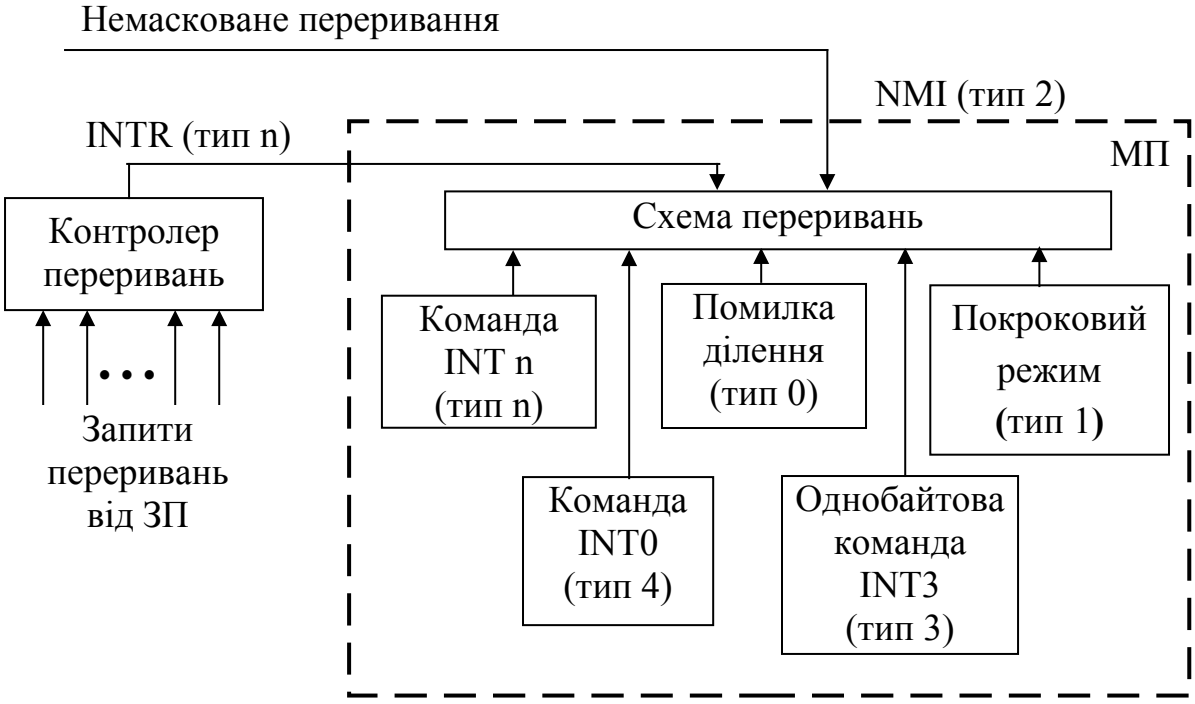


Рисунок 7.13 – Джерела переривань

Зовнішні переривання можуть виникати як реакція на запит переривань від зовнішніх пристроїв, вони є апаратні. Запити на масковані переривання надходять від периферійних пристроїв на контролер переривань, який формує вхідний сигнал МП INTR. У відповідь МП, якщо переривання дозволені програмно, видає сигнал підтвердження переривань INTA, після чого контролер посилає у МП по шині даних байт, який визначає тип переривання і МП його обробляє. Якщо програмно переривання заборонені, МП ігнорує запит, і продовжує виконання програми.

Запити на немасковане переривання надходять по входу NMI і використовуються для переривання роботи МП у таких випадках, як аварійне вимкнення живлення, помилка пам’яті тощо. Реакція на немасковані переривання не залежить від їх програмного дозволу. У більшості випадків МП

реагує на запит будь-якого переривання після закінчення поточної команди. Винятком є строкові команди та команда WAIT, які можуть бути перервані під час виконання.

Пріоритети переривань наведені у табл. 7.2.

Таблиця 7.2 – Пріоритети переривань

Вид переривання	Тип переривання	Пріоритет	Час виклику підпрограми переривань
За командою “помилка ділення”	0	1	50
За командою INTn	5-31	1	51
За командою INT0	4	1	53
За командою INT3	3	1	52
По входу NMI	2	2	50
По входу INTR	32-255	3	61
За прапорцем TF	1	4	50

Оскільки зовнішні пріоритети можна маскувати, це може призвести до перерозподілу призначених різним перериванням пріоритетів. Немасковані переривання, як такі, що мають найвищий пріоритет, можуть маскувати зовнішні до закінчення обслуговування немаскованого переривання. Обслуговування запиту маскованого переривання може бути дозволено програмно шляхом установлення прапорця IF.

Внутрішні переривання можуть бути апаратними та програмними.

Переривання через помилку ділення (тип 0) генеруються МП відразу після виконання команд ділення, якщо формат частки перевищує формат приймача, або при діленні на нуль.

Покрокове переривання (тип 1) виробляється автоматично при установленні прапорця TF після виконання кожної команди, яка змінює вміст сегментного регістра, або у режимі налагодження. Підпрограма обробки покрокового переривання здійснює індикацію внутрішніх регістрів МП та деяких комірок пам’яті.

Переривання за командою INT3 реалізуються у вказаних точках програми.

Переривання за переповненням (тип 4) генерується після виконання команди INTO, якщо установлено прапорець OF.

Переривання, яке визначається користувачем при складанні програм, здійснюється двобайтовою командою INTn.

Програмні переривання дозволяють процедурам викликати одна одну через таблицю вказівників векторів (адрес) переривань, яка розміщена у пам’яті фіксовано, починаючи з адреси 00000H до адреси 003FFH, і займає 1 кбайт пам’яті (рис. 7.14).

Кожний елемент таблиці вміщує два слова, які визначають початкову логічну адресу підпрограми. Слово з більшою адресою вміщує базову адресу сегмента, а слово з меншою адресою – зміщення підпрограми від початку кодового сегмента. При переході на підпрограму адреса сегмента

завантажується у регістр CS, а зміщення – у регістр IP. Розмір кожного елемента таблиці складає 4 байти, МП обчислює адресу кожного елемента шляхом множення потрібного n на 4. При адресуванні елементів таблиці значення жодного сегментного регістра не використовується. На початку при виконанні підпрограм за командою INT*n* зовнішні переривання забороняються, але сама підпрограма може їх дозволити. Немасковані та внутрішні переривання можуть перервати виконання підпрограми, але у ній не припустимі переривання того типу, який вона обслуговує.

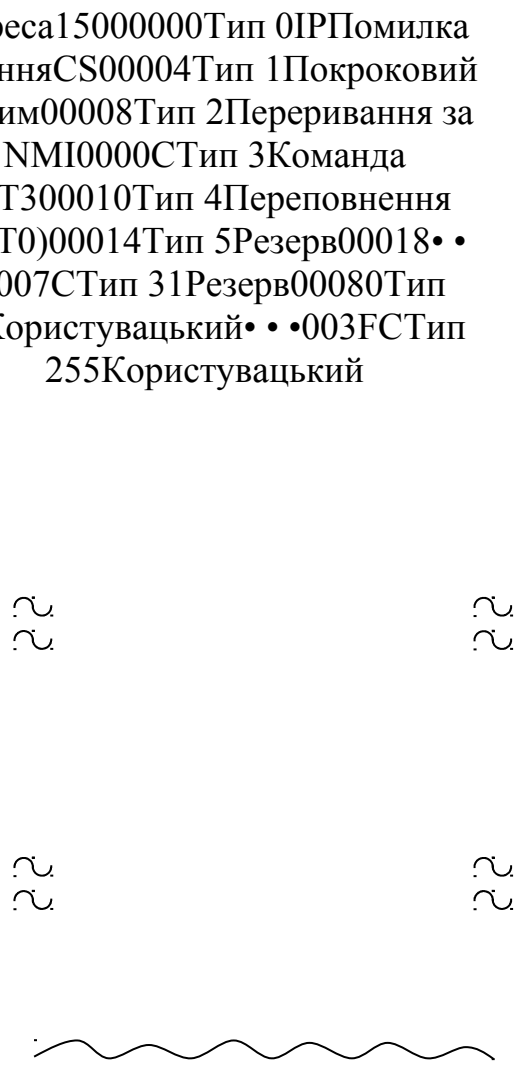


Рисунок 7.14 – Таблиця вказівників векторів переривань

Внутрішні апаратні переривання у більшості моделей мікропроцесорів обумовлені перш за все роботою системного таймера.

Контрольні запитання:

1 Які типи переривань підтримує МП I8086?

2 Де зберігаються вказівники векторів переривань у МП I8086?

3 До якого типу переривань призведе вимкнення живлення або помилка пам’яті?

Контрольні запитання підвищеної складності:

1 Визначіть фізичну адресу сегмента таблиці переривань типу 20.

2 З якою метою забороняються зовнішні переривання на початку виконання підпрограм оброблення переривань?

7.2.5 Організація 32-розрядних мікропроцесорів (Для самостійного вивчення)

Вхідний контроль:

1 Прослідкуйте розвиток МП фірми Intel та поясніть, завдяки яким рисам архітектури МП цієї фірми вийшли на перше місце з продажу на ринку.

2 З якою метою у МПС на МП фірми Intel використовуються співпроцесори?

Перший повністю 32-розрядний МП фірми Intel МП I80386 був створений у 1985 р., для нього було придатне існуюче у 80-ті роки минулого століття прикладне програмне забезпечення вартістю 6,5 млрд. дол., написане для МП попередніх моделей від I8086 до I80286, тобто зберігалась сумісність знизу вверх. МП I80386 підтримує багатозадачність, вбудоване керування пам'яттю, віртуальну пам'ять, захищений режим. МП може виконувати до 4 млрд. операцій за секунду, працює під керуванням операційних систем MS-DOS та UNIX і припускає роботу з співпроцесорами I80287 та I80387. Пам'ять сегментується, розмір сегмента може становити від 1 байта до 4 Гбайт.

Представником другого покоління 32-розрядних мікропроцесорів є МП I80486, який з'явився у 1989 р. Порівняно з МП I80386 він має такі ознаки:

– на його кришталі додатково розміщені кеш-пам'ять даних та команд, процесор обробки даних з плаваючою точкою. Найбільш часто використовувані команди виконуються за один цикл. МП I80486 має апаратну та командну підтримку роботи у складі багатопроцесорних систем і зовнішню кеш-пам'ять другого рівня. Продуктивність мікропроцесора забезпечується на рівні МП архітектури RISC. Модель МП I80486DX2 працює на частоті 66 МГц.

Слід зазначити, що деякі засоби телекомунікацій, зокрема, ЦАТС Квант-Е використовують 32-розрядні МП другого покоління як керувальні пристрої для реалізації комутаційних полів.

МП I80486 може працювати з операційними системами MS-DOS, Microsoft Windows, Unix System V/386, операційною системою реального часу *i*RMX.

З появою 32-розрядних процесорів найбільш перспективною стає сегментована захищена модель, в якій пам'ять складається з захищених незалежних сегментів, які можна цілком переміщати з жорсткого диску в ОЗП і навпаки. Кожній з програм у багатозадачному режимі надаються сегменти стека, коду та до 4-х сегментів даних. Некоректні звернення додатків до пам'яті блокуються системою захисту, а для запису та читання кодів як даних потрібні деякі штучні засоби – перевизначення сегментів.

Мікропроцесор I80386 призначений для виконання 32-розрядних операцій, але він може функціонувати як швидкі МП I8086 та МП I80286. МП I80386 може працювати у реальному, захищеному режимах та режимі віртуального I8086. Різниця між цими режимами полягає у способах адресації та обсязі адресованої пам'яті. У реальному режимі МП I80386 працює як 16-розрядний I8086 і може адресувати пам'ять обсягом 1 Мбайт. У захищеному режимі МП працює як 16-розрядний I80286 та адресує пам'ять 1 Гбайт або як 32-розрядний з можливістю адресування пам'яті 4 Гбайти; можлива також організація пам'яті зі сторінковою підтримкою та апаратна підтримка захисту пам'яті від несанкціонованого доступу, реалізуються принципи захищеного режиму адресації пам'яті.

Коли виконуються команди з даними 32-розрядної довжини, МП I80386 використовує 8- або 32-розрядні зміщення і будь-який регістр, крім ESP, може бути використаний як базовий або індексний. При виконанні команд з даними 16-розрядної довжини як базові регістри використовуться регістри BX, BP, індексні – SI, DI, зміщення може становити 0, 8, 16 біт.

Програмна модель МП I80386 вміщує регістри загального призначення EAX, EDX, ECX, EBX, регістри-вказівники ESP, EBP, ESI, EDI, сегментні регістри CS, SS, DS, ES, FS, GS, вказівник команд EIP, регістр прапорців EFLAGS.

РЗП вміщують підмножину регістрів МП I8086, мають довжину 32 розряди і можуть вміщувати адреси та дані. В них зберігаються дані довжиною 1, 8, 16, 32 або при використанні двох регістрів 64 біти, або розрядні поля від 1 до 32 біт, адреси довжиною 16 або 32 біт. Програмна модель МП показана на рис. 7.15.

Рисунок 7.15 – Регістри загального призначення

РЗП називаються EAX, EBX, ECX, EDX, ES; регістри-вказівники – EBP, ESP; індексні регістри – ESI, EDI. Доступ до молодших 16 розрядів цих регістрів виконується при використанні імен 16-розрядних регістрів AX, BX, CX, DX, SI, DI, BP та SP. Розширений вказівник команд (EIP) є 32-розрядний регістр. Він вміщує відносну адресу наступної виконуваної команди. Доступ до

молодших 16 розрядів регістра EIP здійснюється при використанні 16-розрядного імені регістра IP. Це може знадобитися при виконанні програм для мікропроцесорів I8086 та I80286, які мають тільки регістр IP. Регістр прапорців EFLAGS керує введенням-виведенням, перериваннями, які можна маскувати, налагодженням програм та систем, перемиканням задач, включенням та виконанням задач МП I8086 у віртуальному режимі у захищеному багатозадачному середовищі.

Молодші 16 розрядів регістра EFLAGS є 16-розрядний регістр прапорців з назвою FLAGS. Регістр EFLAGS показано на рис. 7.16.

Рисунок 7.16 – Регістр вказівника команд та прапорці

Прапорці результату, які вміщуються у регістрі FLAGS:

CF(0) – прапорець перенесення із старшого розряду. CF установлюється, якщо у результаті виконання операції є перенесення (при складанні) або зайом (при відніманні) відповідно зі старшого або у старший розряд. Для 8-, 16-, 32-розрядних операцій CF установлюється відповідно при перенесенні з розрядів 7, 15, 31.

AF(4) – прапорець допоміжного перенесення. AF установлюється, якщо у результаті виконання операції додавання чисел в упакованому форматі BCD виникає перенесення з розряду 3 незалежно від довжини операндів – 8, 16 або 32 розряди.

OF(11) – прапорець переповнення, установлюється, якщо у результаті виконання операції виникає знакове переповнення.

ZF(6) – прапорець нуля, установлюється, якщо результат операції дорівнює 0.

SF(7) – прапорець знака, установлюється, якщо установлено старший розряд результату, для 8-, 16- та 32-розрядних операндів SF співпадає зі значенням розрядів 8, 16, 31.

PF(2) – прапорець паритету (парності), установлюється, якщо молодший байт результату операції вміщує парну кількість одиниць.

Прапорці керування:

DF(10) – прапорець напряму, показує напрям проходження операндів при виконанні операцій з рядками (0 – у напрямі збільшення адрес, 1 – у напрямі зменшення адрес, при адресуванні за участі індексних регістрів SI та DI).

IF(9) – прапорець переривань, дозволяє (1) сприйняття переривань, які надходять з входу INTR.

IOPL(12, 13) – прапорець привілеїв, вказує на максимальне значення поточного рівня привілеїв (CPL).

TF(8) – прапорець трасування, при TF = 1 МП переводиться у покроковий режим для налагодження програми.

NT(14) – прапорець вкладності задач, установлюється в 1, якщо вирішувана задача вкладена у іншу задачу; значення NT у EFLAGS перевіряється при виконанні команди IRET для визначення, звідки треба вибрати адресу повернення:

при NT = 0 IRET вибирає адресу з поточного стека;

при NT = 1 IRET вибирає адресу із сегмента стану поточної задачі (TSS).

VM(17) – прапорець режиму, VM установлюється в 1 тільки у захищеному режимі при бажанні включення віртуального I8086.

RF(16) – прапорець відновлення, при RF = 1 використовується режим налагодження з точками зупину.

Сегментні регістри:

Шість 16-розрядних сегментних регістрів вміщують значення селекторів сегментів, які у реальному режимі адресації вказують на поточні сегменти адрес пам’яті: CS – сегментний регістр кодів команд програми, регістр сегмента стека SS і чотири регістри сегментів даних DS, ES, FS, GS (рис. 7.17).

Лінійна адреса операнда або команди є сума базової адреси сегмента та 32-розрядної відносної адреси.

Для адресації пам’яті у захищеному режимі в МП I80386 є “невидимі” для програм користувача регістри дескрипторів (описувачів) сегментів, в яких зберігається інформація, яка керує пам’яттю. До 16-розрядних регістрів МП I80286 додані 64-розрядні дескриптори сегментів. МП I80386 вибирає дескриптори сегментів з таблиць дескрипторів, використовуючи сегментний регістр як індекс при зверненні до цієї таблиці.

МП I80386 перетворює логічні адреси (використані програмістом у програмі) у фізичні адреси двома способами:

– перетворенням сегмента, коли логічна адреса (вміст селектора сегмента та зміщення сегмента) перетворюється у лінійну адресу, яка і є фізичною;

– перетворенням сторінки, коли лінійна адреса перетворюється у фізичну адресу.

Обидва способи є скриті від програміста і підтримуються апаратно.

Рисунок 7.17 – Сегментні регістри

На рис. 7.18 показано узагальнену схему перетворення адрес у МП I80386.

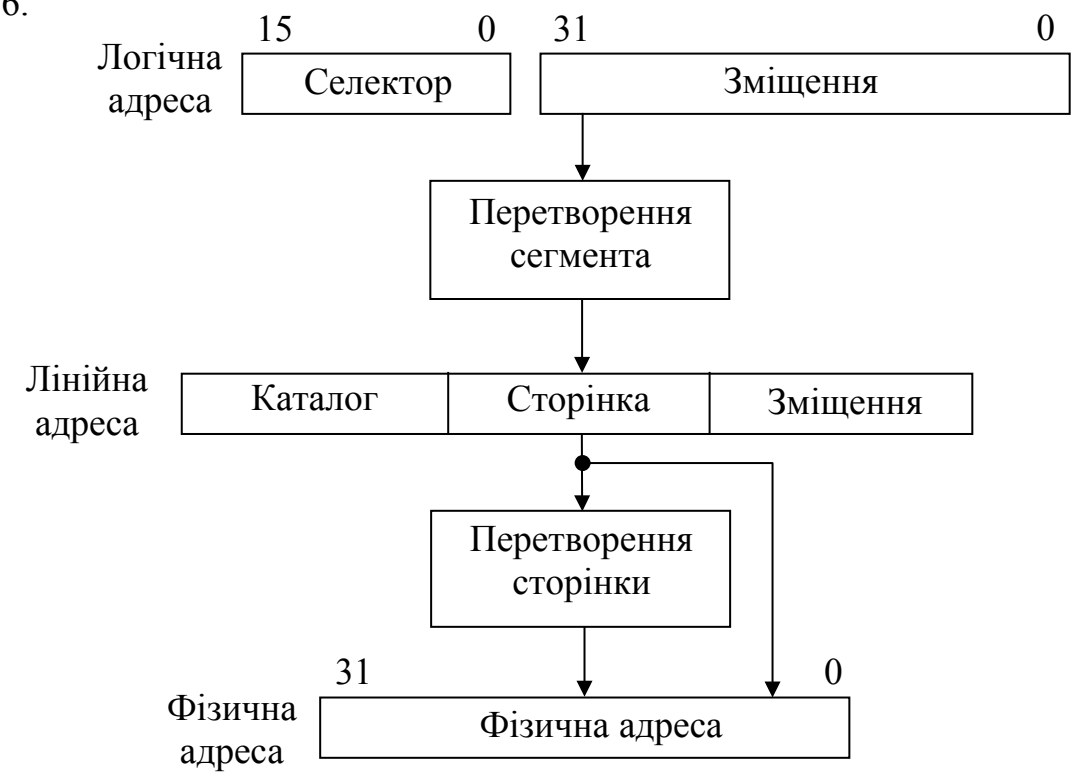


Рисунок 7.18 – Процес перетворення адреси в МП 80386

На рис. 7.19 показано, як перетворюється вміст сегментного регістра та зміщення (логічна адреса) на лінійну адресу – перша фаза.

При сегментній організації пам’яті цієї фази достатньо для отримання фізичної адреси комірки пам’яті.

При сторінковій адресації потрібно виконати ще одну фазу перетворення – лінійна адреса перетворюється у фізичну.

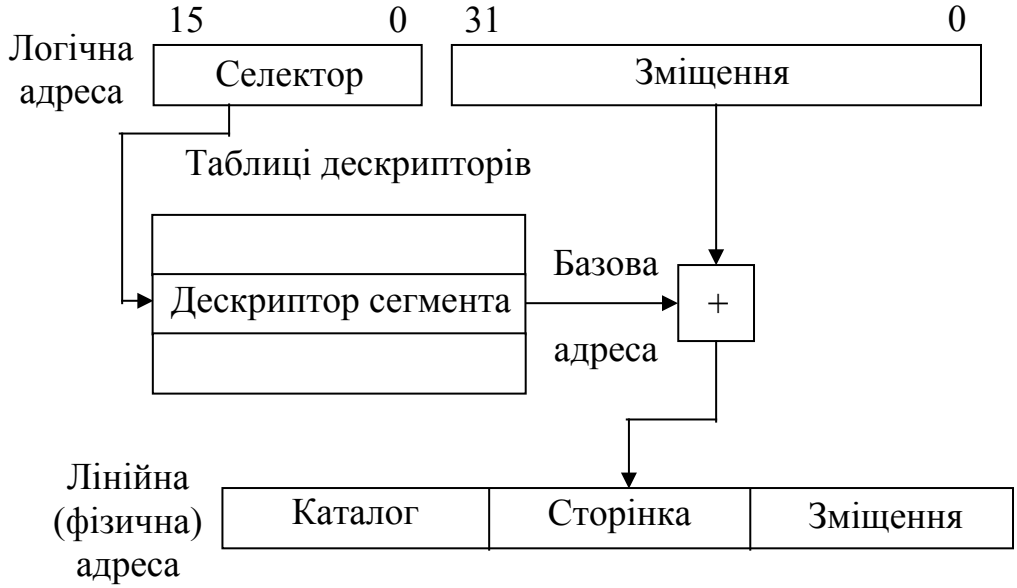


Рисунок 7.19 – Сегментна організація пам’яті

При сторінковій організації пам’яті (рис. 7.20) лінійну адресу можна трактувати як складену з трьох полів: каталогу, сторінки та зміщення. На рис. 7.20 показано дворівневу систему. Поле “Каталог” вказує на таблицю сторінок, яка відноситься до конкретного елемента каталогу, а поле “Сторінка” використовується як індекс у таблиці сторінок, які визначаються каталогом; поле “Зміщення” використовується як адреса байта всередині сторінки, вказаної

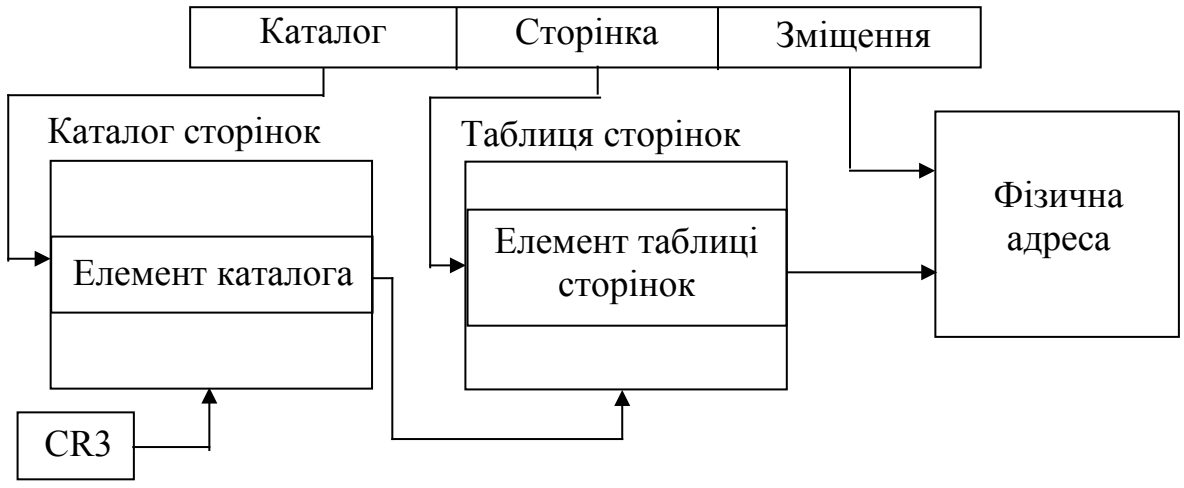


Рисунок 7.20 – Перетворення сторінок

таблицею сторінок, яка є область 1К 32-розрядних елементів. Для адресації пам’яті використовуються два рівня таблиць. На вищому рівні знаходиться каталог сторінок, який адресує до 1К сторінок таблиць другого рівня. Таблиця сторінок другого рівня адресує до 1 кбайта сторінок другого рівня. Вся таблиця

адресується одною сторінкою каталогу, тому у сукупності можна адресувати 1 Мбайт сторінок. Розмір однієї сторінки має 4 кбайти, таким чином, таблиці одного каталогу сторінок можуть адресувати всю область фізичних адрес МП I80386. Фізична адреса поточного каталогу сторінок зберігається у регістрі CR3 МП. Програми керування пам'яттю можуть задати один каталог для всіх задач, по одному каталогу сторінок на кожну задачу або комбінації з кількох каталогів. Для підвищення продуктивності при перетворюванні адрес МП зберігає найбільш часто використовувані адреси таблиці сторінок у внутрішній кеш-пам'яті, яка має назву буфер швидкої переадресації. Якщо у кеш-пам'яті немає сторінкової інформації, треба звертатися до обох рівнів таблиць сторінок. Керувати кеш-пам'яттю може тільки системний програміст.

Слід зазначити, що при сегментній адресації пам'яті сегмент може або цілком знаходитись у пам'яті, або ні; при значних розмірах сегментів це призводить до перевантаження МП; сторінковий спосіб адресації є більш ефективним через невеликий розмір сторінки (4 кбайти) та можливість часткового знаходження на жорсткому диску та у пам'яті.

МП I80386 має чотири рівні захисту привілеїв – для ізолювання програм користувача одна від одної і кожної з них від операційної системи і навпаки у багатозадачному режимі. Правила доступу до даних такі:

– дані, які зберігаються у сегменті з рівнем привілеїв *p* можуть бути доступними тільки при виконанні команд з такими самими або меншими привілеями, ніж *p*;

– кодовий сегмент з рівнем привілеїв p може бути викликаний задачею, виконуваною на тому самому рівні або більш привілейованому, ніж *p*.

МП I80386 здійснює контроль типу дескриптора сегмента, який виконується для пошуку програмних помилок при спробі використати сегменти у випадках, коли це не передбачено системним програмістом. МП перевіряє тип сегмента у двох випадках: коли селектор дескриптора завантажується у сегментний регістр і коли команда звертається до сегментного регістра, причому жодна команда не може нічого записувати у виконуваний сегмент, неможливе читання кодового сегмента, який має доступ тільки на виконання, неможливий запис у сегмент даних, який має доступ тільки на читання, контролюється межею сегмента. При виконанні команд керування JMP, CALL та RET передача управління здійснюється всередині поточного кодового сегмента і тому контролюються тільки межею сегментів.

При міжсегментних переходах команди JMP і CALL можуть звертатись до інших сегментів, тому МП здійснює контроль привілеїв сегментів і дозволяє перехід безпосередньо між сегментами, або через шлюзи при виконанні певних умов.

При виконанні команди RET, якщо керування на повернення з процедури є внутрішньосегментним, межі поточного сегмента не контролюються. Міжсегментна форма команди RET відновлює вказівник стека, і МП перевіряє привілеї сегментів, тому що є вірогідність пошкодження або зміни поточного стека.

Співпроцесори МП І80386

МП І80386 може працювати з арифметичними співпроцесорами І80287 і І80387 та співпроцесором локальної обчислювальної мережі І82586.

Якщо МП І80386 працює з співпроцесором І80387, то він виконує 32-розрядні передачі. Робота співпроцесора з процесором реалізована як синхронна або псевдосинхронна.

Співпроцесор локальної обчислювальної мережі (ЛОМ) є високопродуктивним інтелектуальним контролером зв'язку, здатним вирішувати задачі керування доступом до локальної обчислювальної мережі. На рис. 7.21 показано структурну схему робочої станції локальної обчислювальної мережі Ethernet.

Рисунок 7.21 – Структурна схема робочої станції ЛОМ

Співпроцесор виконує усі функції, які відносяться до передавання даних між розподілюваною пам'яттю та каналом мережі, а саме фільтрацію адреси, керування каналом, розбиття даних на кадри, виявлення помилок, кодування даних, керування мережею, прямий доступ до пам'яті, організація буферної пам'яті.

Робоча станція вміщує центральний процесор I80386, послідовний інтерфейс, розподілювану пам’ять, приймач-передавач та канал Ethernet.

Кеш-пам'ять МП I80386 розташована між основною пам’яттю і процесором для збільшення швидкодії системи. Механізм кеш-пам’яті є прозорим для програміста, а саме слово кеш є тайник. Швидкодія кеш порівняна з швидкодією основної пам’яті.

Кеш-пам'ять МП I80386

Концепція кеш-пам’яті основана на передбаченні найбільш вірогідного використання процесором даних з основної пам’яті шляхом копіювання їх у кеш. Ця концепція розповсюджується на дані, сусідні з поточними даними. Для цього зазвичай здійснюється передавання з основної пам’яті у кеш-пам'ять блока з кількох слів, навіть якщо у даний момент потрібне тільки одне слово. Якщо потрібне слово є частиною потоку послідовних команд, то вибирання наступних команд виконується на першому етапі роботи, оскільки завдяки цьому зникає необхідність повторного звернення до основної пам’яті. Коли процесору потрібно зчитати або записати дані у пам'ять, то спочатку він перевіряє їх наявність у кеш. Якщо дані знаходяться у кеш-пам’яті (кеш-попадання), то МП I80386 може їх швидко отримати. У разі їх відсутності (кеш-промах) ці дані зчитуються з основної пам’яті та переписуються у кеш. МП I80386 працює з зовнішньою кеш-пам’яттю даних та команд.

Передбачення адреси наступного звернення до пам’яті можливе тому, що програми звертаються до пам’яті за адресою, близькою до адреси попереднього звертання. Цей принцип називається локальністю програми. Для збільшення відсотка кеш-попадань процесор використовує блочну вибірку. Контролер кеш-пам’яті поділяє на блоки основну пам’ять довжиною 2, 4, 8 або 16 байт. Зазвичай 32-розрядний процесор вміщує два або чотири слова на блок. У разі промаху контролер пересилає увесь блок, який вміщує потрібне слово, з основної пам’яті у кеш. Блочна вибірка дозволяє записати у кеш дані, розташовані перед потрібним байтом (перегляд назад), або дані, розташовані після нього (перегляд вперед) або обидва випадки.

Кеш-пам'ять у МП I80386 має ємність 64 кбайт з 16384 входами. Кожний вхід вміщує 32 біти даних разом з 16 бітами адресної інформації. Довжина рядка (одиниця обміну між МП та кеш) складає 32 біти. Для керування пам’яттю використовуються два компаратори та дві програмовані логічні матриці. Сама ж пам’ять складається з восьми статичних ОЗП даних з організацією 8К х 8 і два динамічних ОЗП тегового поля з організацією 8К х 8. Кеш-пам’ять побудована за принципом прямого відображення, кожний елемент в основній пам’яті відображається в один елемент усередині кеш. 30-розрядна фізична адреса процесора поділяється на дві частини: 16-розрядний тег та 14-розрядний індекс. Тег відповідає старшій адресі рядків (А16 – А31), а індекс – молодшій адресі рядків (А2 – А15). Поля тега та індексу разом визначають місце подвійного слова в основній пам’яті. Додатково індекс вибирає один з 16384 рядків у кеш. Наймолодші два біти адреси кодуються так, що

доступними є всі чотири байти подвійного слова. Кеш-пам'ять використовує буферізований наскрізний запис для оновлення динамічного ОЗП. Під час циклів запису нові дані записуються у кеш-пам’ять і одночасно в ОЗП. При використанні буфера цикл запису в ОЗП перекривається за часом з наступним циклом магістралі, що дозволяє виконувати запис за три такти. Буферізований наскрізний запис реалізується шляхом відділення режиму роботи ПК з кеш від режиму роботи з ДОЗУ. Це можливо тому, що кеш та динамічний ОЗП мають кожний свої власні контролери, що забезпечує можливість операцій з перекриттям цих функцій. Керувальна програмована логічна матриця визначає, коли у режимі роботи з кеш необхідний цикл звернення до пам’яті, а у режимі роботи з динамічним ОЗП запит перевіряється тільки у разі готовності до запуску нового циклу.

При оновленні динамічного ОЗП будь-який запис буферізується, тобто інформація затримується у кеш-пам’яті перед записом в основну пам’ять, а схеми кеш керують процесом доступу до основної пам’яті асинхронно по відношенню до роботи процесора. Процесор починає новий цикл до завершення циклу запису в основну пам’ять. Якщо за записом відбувається зчитування, то обов’язково буде кеш-попадання, тому що читання виконується у той час, коли контролер кеш зайнятий оновленням основної пам’яті. Буферізація підвищує продуктивність кеш. На рис. 7.22 наведено приклад буферізованої наскрізної пам’яті. Припустимо, що МП I80386 закінчив запис “нових даних 1”. Контролер кеш розмістив у буфер дані для запису й оновлення кеш. Процесор зчитує “старі дані 2” з кеш. Виникає ситуація кеш-попадання. У той час, поки відбувається читання “старих даних 2”, “нові дані 1” переписуються в основну пам’ять замість “старих даних 2”.

Недоліком буферізованого наскрізного запису є буферизація поодинокого запису, через що два останніх записи в основну пам’ять потребують циклу чекання процесора. Запис перепущеним наступним читанням потребує стану чекання процесора, який використовується для синхронізації його з повільно діючою пам’яттю. Це знижує продуктивність системи.

Останні моделі МП I80486 та подальші моделі використовують більш продуктивний режим зворотного запису, за наявності якого дані записуються в ОЗП тільки тоді, коли вони змінюються.

Контрольні запитання:

1 Які системні функції підтримують 32-розрядні МП I80386 та I80486?

2 Під якими операційними системами можуть працювати ці МП?

3 В яких режимах працюють МП I80386 та I80486?

4 З якою метою використовується захищений режим?

5 Які переваги забезпечують сегментний та сторінковий способи перетворення віртуальних адрес?

6 Опишіть концепцію кеш-пам’яті.

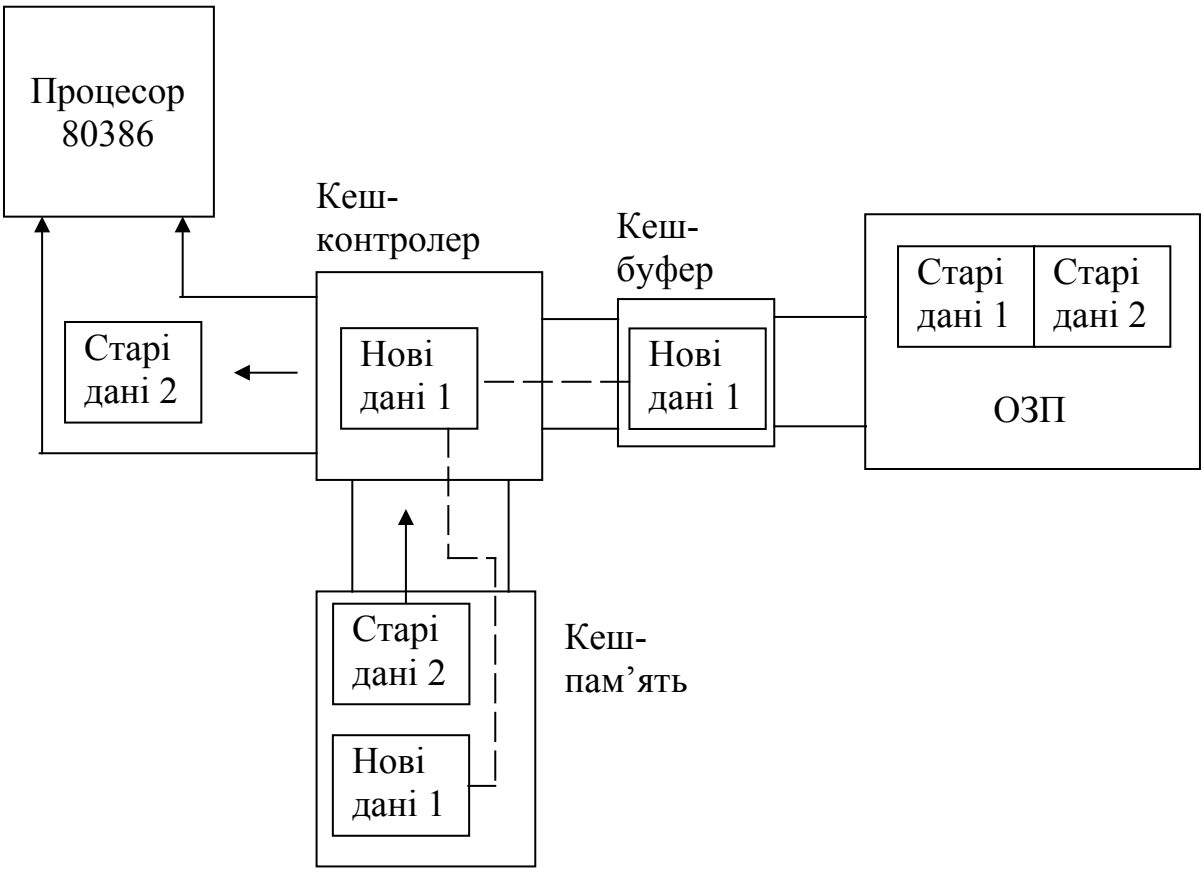


Рисунок 7.22 – Приклад буферізованої наскрізної пам’яті

Контрольні запитання підвищеної складності:

1 Як адресується кеш-пам’ять?

2 Опишіть режим зворотного запису у кеш-пам’ять.

7.3 Архітектура сучасних мікропроцесорів

7.3.1 Тенденції розвитку архітектури сучасних мікропроцесорів

Вхідний контроль:

1 З якою метою у МП вбудовуються конвеєри даних та команд?

2 Чи можна у програмі передбачити виконувану вітку перед виконанням команди умовного переходу?

Сучасний підхід до організації обчислювального процесу на мікропроцесорах має дві тенденції. Перша базується на припущенні, що система команд не вміщує вказівок на паралельну обробку даних всередині процесора, але є апаратна підтримка виконання кількох команд за один такт. Такі процесори відносяться до суперскалярних. Друга передбачає включення до формату команд спеціальних полів, які містять вказівки для кожного з паралельно оброблюючих дані пристроїв. Такі процесори називаються процесорами з **довгим командним словом** (VLIW) і потребують також

наявності відповідних компіляторів з мов високого рівня, які подають програми у машинних кодах для завантаження їх у процесори.

Розвиток суперскалярних мікропроцесорів відбувається шляхом якомога більшої кількості паралельних структур (конвеєрів) при традиційно послідовних програмах. Без втручання програміста апаратні та програмні засоби процесора забезпечують завантаження паралельно працюючих функціональних пристроїв процесора.

З метою підвищення продуктивності апаратно-програмні засоби процесора паралельно виконують кілька команд іноді у порядку, відмінному від їх розташування у програмі. І суперскалярні, і VLIW процесори використовують паралелізм на рівні команд.

Основною перешкодою для високопаралельного виконання програм є залежність з керування (розгалуження), її треба виявити перш, ніж будуть виконані усі наступні команди.

Процес виконання програми з конкретними наборами даних може бути представлений динамічною структурою програми, тобто у тому порядку виконання команд, як його оптимізує процесор разом з транслятором. Обмеженнями при оптимізації є залежність і по командах, і по даних. При виконанні програми процесор просувається по ній за допомогою вікна виконання. Якщо команди, які попали у вікно, є незалежні, то вони можуть виконуватись паралельно. Для усунення залежностей, викликаних командами переходів, використовується метод передбачення, який може викликати у конвеєр команд та умовно виконувати команди передбачених переходів. Якщо передбачення зроблено вірно, то результати виконуваних команд враховуються, якщо ж ні, то стан процесора відновлюється на момент прийняття рішення про виконання команд.

Команди у вікні виконання можуть також бути залежними по даних, що обумовлено використанням тих самих регістрів, комірок пам’яті або результатів попередніх команд. Деякі з цих залежностей можна усунути, використовуючи додаткові ресурси процесора, після чого команди можна виконувати паралельно. Основні блоки суперскалярного процесора – це блок вибірки команд і передбачення переходів, блок декодування команд, аналізу залежності між командами, перейменування регістрів, диспетчеризація , блоки регістрів та оброблюючих пристроїв з плаваючою та фіксованою точками, блок керування пам’яттю, а також блок упорядкування виконаних команд.

Роздільні багаторівневі кеш-пам’яті даних та команд забезпечують одночасне введення у паралельно працюючі функціональні пристрої процесора кількох команд за один такт; у кеш-пам’яті введені засоби передбачення переходів. Одні засоби передбачення застосовують інформацію з двійкового коду команд або вироблену компілятором. Так, певні коди операцій частіше викликають розгалуження ніж інші, або розгалуження більш вірогідне, наприклад, у циклах, або компілятор у процесі перетворення програми у машинні коди виставляє прапорець, який установлює напрямок переходу. Може використовуватись також статистична інформація, отримана під час трасування програми. Інші засоби передбачення використовують інформацію

щодо історії розгалуження, яка запам'ятовується у таблицях розгалужень та передбачень. Якщо передбачення виявилось невірним, результати команд, які були умовно (спекулятивно) виконані, анулюються.

Після декодування команд створюються групи даних щодо виконуваної операції, адрес операндів та адреси розміщення результату. На наступному кроці відбувається відображення логічних ресурсів, яку потребує програма, на фізичні ресурси мікропроцесора. Якщо один логічний ресурс відображається на кілька фізичних ресурсів, кожний з яких відповідає значенню логічної величини в один з моментів послідовного виконання програми, і команда створює нове значення для логічного ресурсу; фізичний ресурс, в якому розміщується це значення, отримує ім'я. Наступні команди, які потім це значення використовують, сповіщаються відносно імені фізичного ресурсу; частіше за все це стосується перейменування регістрів. Цей прийом усуває залежність різних команд від даних.

У суперскалярних процесорах послідовні програми розбиваються на фрагменти, в яких команди виконуються паралельно. Після заповнення конвеєра командами між ними установлюються тільки необхідні залежності по даних. Ефективне використання суперскалярних процесорів обмежується двома обставинами. По-перше, через наявність умовних переходів паралелізм обчислювань на рівні команд є обмежений. Розмір вікна виконання також обмежує можливий притаманний програмі паралелізм, тому що зовсім не розглядається паралельне виконання команд, які знаходяться за межами вікна. По-друге, складність суперскалярного процесора зростає швидше, ніж кількість паралельно виконуваних команд.

Процесори VLIW використовують задання у командному слові сукупності паралельно виконуваних команд. Підготовка таких програм виконується компілятором. Перевагою VLIW є наявність програмних засобів, які більш ефективно, ніж апаратні, обмежені розміром вікна виконання, можуть аналізувати залежності між командами і вибирати паралельно виконувані команди. Крім того, що є важливим, VLIW-процесор має більш простий пристрій керування і потенціально може мати більш високу тактову частоту. Недоліком VLIW є також обмеження його продуктивності за рахунок обмеженості вікна виконання, вже реалізованого програмно, та проблеми з умовними переходами. Представниками процесорів типу VLIW є процесори фірми Transmeta (МП типу Crusoe моделей TM3120, TM5400, TM5600 з тактовою частотою 100 МГц), Intel (МП Itanium, 800 МГц) і Heulett-Packard – модель McKinley.

Перспективними є також мультискалярні процесори, які розбивають послідовний потік команд на задачі, і забезпечують більшу глибину передбачення та високу вірогідність вибору правильного напрямку обчислень, а також більш широке вікно виконання. Мультискалярний процесор схожий на багатопроцесорну систему з розподілюваною пам'яттю і не вимагає ніяких апріорних знань щодо зв'язків команд з керування та даним. Розпаралелювання задач потребує використання компіляторів з мов високого рівня, які їх розпаралелюють. На ринку мікропроцесорів суперскалярні мікропроцесори є

лідерами. При виборі цих процесорів для розв'язання задач у проблемній області, для якої створюється обчислювальна система, слід перевіряти адекватність прийомів підвищення продуктивності розв'язуваним задачам.

Архітектура 64-розрядних процесорів не є ані 64-розрядним розширенням архітектури CISC, ані переробленням архітектури RISC. Це нова архітектура, яка для забезпечення більшого паралелізму виконання команд використовує довгі слова команд (LIW), предикати команд, попереднє завантаження даних, сумісність на етапі декодування команд VLIW та CISC. Для цього операційні системи повинні мати підтримку як 64-розрядних додатків, так і 32-розрядних.

Контрольні запитання:

1 Які тенденції розвитку архітектури сучасних МП можна назвати?

2 З якою метою процесори можуть виконувати команди у порядку, відмінному до їх написання?

3 Залежність між якими складовими програми заважає оптимізації порядку виконання команд у програмі?

4 Які методи усунення залежностей, викликаних командами переходів, Вам є відомі?

5 З якою метою у сучасних процесорах використовується вікно виконання?

6 Скільки регістрів загального призначення вміщують регістрові файли сучасних процесорів?

7 Які переваги мають суперскалярні процесори?

8 Які переваги мають процесори VLIW?

Контрольні запитання підвищеної складності:

1 З точки зору програміста, для яких процесорів, суперскалярних або VLIW, складніше створювати програмне забезпечення і чому?

2 Що таке потоковий процесор?

3 Що таке мультискалярний процесор?

4 З якими операційними системами можуть працювати 64-розрядні процесори?

7.3.2 Мікропроцесори Pentium

Вхідний контроль:

1 Скільки інструкцій виконував МП I80486 за один такт?

2 Скільки конвеєрів мав МП I80486?

Мікропроцесори п'ятого покоління Pentium принципово відрізняються від I80486 своєю суперскалярною архітектурою – здатністю виконувати на своїх конвеєрах до двох інструкцій при частоті 200 МГц. МП Pentium повністю програмно сумісний з попередніми МП Intel і дозволяє використовувати раніш розроблене програмне забезпечення для ПК.

Технічні новації, притаманні мікропроцесору, є також такі:

– окремі кеш-пам’яті для команд та даних;
– передбачення переходів;
– високопродуктивні операції з плаваючою точкою;
– удосконалена 64-розрядна шина даних;
– засоби забезпечення цілісності даних;
– засоби керування енергоживленням;
– підтримка багатопроцесорності;
– моніторинг продуктивності;
– підтримка сторінкової пам’яті різних розмірів.

Структурна схема МП Pentium показана на рис. 7.23.

Два конвеєри команд *U* та *V* паралельно можуть виконувати дві різні команди. Конвеєр *U* виконує усі команди, а *V* – їх обмежений набір більш простих команд.

Кожна кеш-пам'ять процесора Pentium має розмір 8 кбайт. Буфер трансляції адрес (TLB) перетворює адресу комірки зовнішньої пам’яті у відповідну адресу даних у кеш. У МП Pentium використовується метод зворотного запису, коли дані записуються в ОЗП тільки при їх видаленні з кеш. Це дозволяє модифікувати дані у кеш без збереження в ОЗП. При роботі МП Pentium у багатопроцесорній системі завдяки протоколу MEST (Modified, Exclusive, Shared, Invalid) забезпечується узгодженість даних в усіх кеш мікропроцесорів системи і в основній пам’яті.

Передбачення переходів у програмах реалізується завдяки буферу BTB (Branch Target Buffer) і двом буферам попередньої вибірки. Один із них використовується для попередньої вибірки команди у припущенні, що переходу не буде, а інший виконує передвибірку інструкцій у буфер, використовуючи вміст BTB (запам’ятовуване при першому виконанні переходу). Такий алгоритм не тільки прогнозує вибір простих віток, але й виконує складне передбачення в укладених циклах, завдяки тому, що в буфері BTB можуть зберігатись кілька (до 256) адрес переходів.

У МП Pentium застосовується блок обчислень з плаваючою точкою, який використовує складні багатоступеневі конвеєри та внутрішні функції. Майже всі команди з плаваючою точкою починають виконуватись у конвеєрі *U* або *V*, а потім передаються на конвеєр з плаваючою точкою, тому цілочислові конвеєри не є незалежні – при зупинці одного з них для роботи з числами з плаваючою точкою другий теж зупиняється. Такі функції, як множення, ділення, додавання реалізовані як внутрішні у просторі операцій з плаваючою точкою.

Всі ці нововведення призводять до перевищення швидкості виконання операцій з плаваючою точкою у 10 разів порівняно з МП I80486 DX 33 МГц. Завдяки 64-розрядній шині даних МП Pentium може обмінюватись даними з пам’яттю зі швидкістю 528 Мбайт/с. МП Pentium реалізує конвеєризацію циклів шини, що дозволяє почати другий цикл ще до завершення першого. Для підвищення швидкості виконання послідовних операцій запису в пам’ять МП Pentium має по одному буферу запису на кожний конвеєр, завдяки чому

мікропроцесор може продовжувати роботу, виконуючи наступні команди до запису результату попередньої команди в пам’ять.

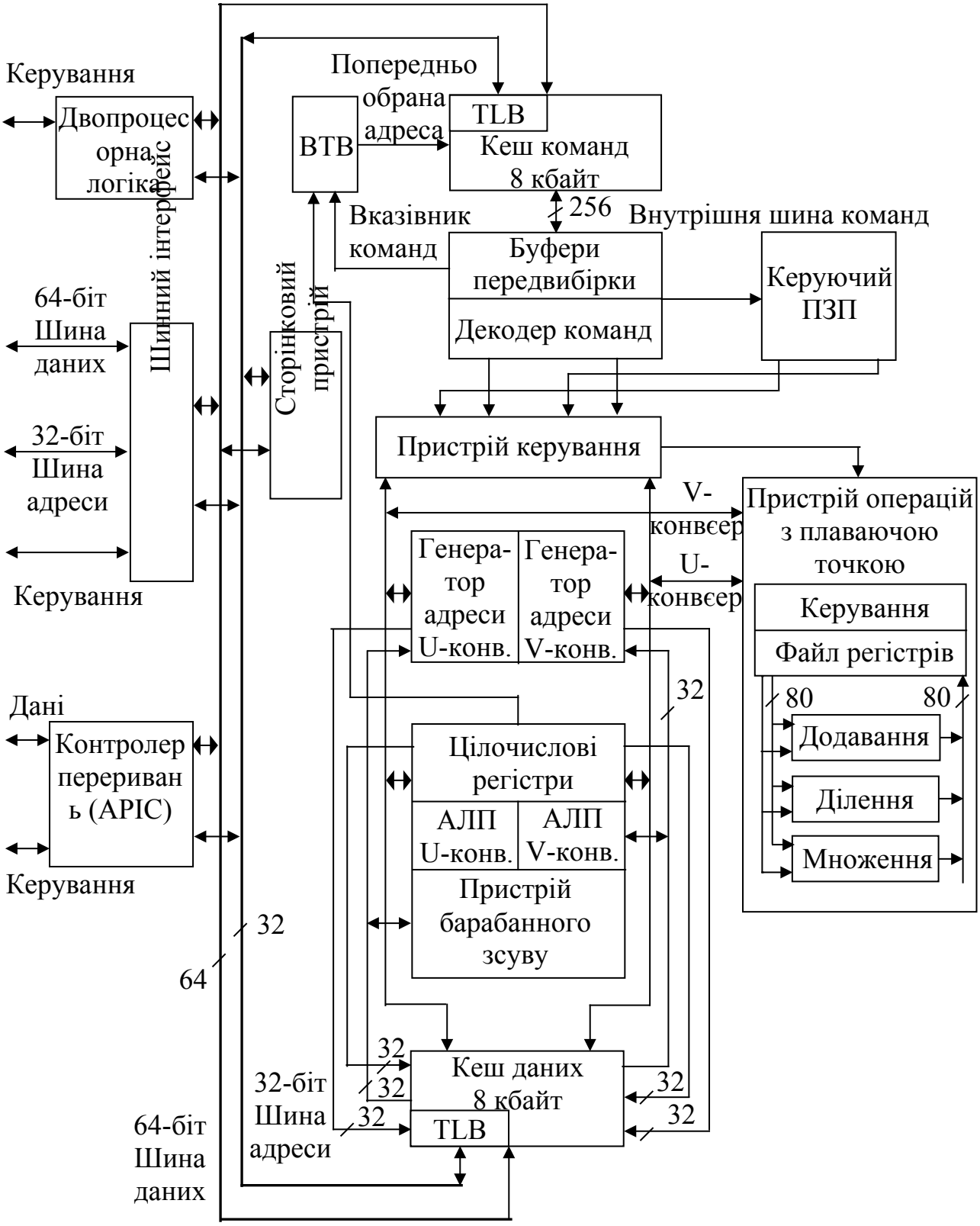


Рисунок 7.23 – Структурна схема МП Pentium

Цілісність даних перевіряється за ознакою парності одиниць у внутрішніх буферах процесора.

У МП Pentium засоби енергозбереження застосовуються як на рівні мікропроцесора, так і на рівні системи. При виконанні задач, які не потребують інтенсивних обчислень, процесор може бути переведений у режим зі зниженою тактовою частотою та зниженим енергоживленням.

Підтримка багатопроцесорності забезпечується наявністю внутрішнього контролера багатопроцесорних переривань APIC, який підтримує до 60 процесорів у системі, та двовходового контролера кеш-пам'яті другого рівня, який дозволяє двом процесорам сумісно використовувати один кеш другого рівня.

Мікропроцесор Pentium підтримує розміри сторінки пам'яті 4 кбайти та 4 Мбайти. Це дає можливість регулювати частоту переключення сторінок у ядрі операційної системи або у графічних додатках.

МП Pentium MMX характеризується апаратною підтримкою багатьох операцій, характерних для процесорів цифрової обробки сигналів: одна інструкція виконує дії над кількома (2, 4, 8) комплектами операндів (принцип SIMD), операції над векторами, згортка, перетворення Фур'є тощо. Це стало можливим завдяки розширенню системи команд на 57 команд, орієнтованих на ефективне виконання типових мультимедійних алгоритмів. Процесор вирішує задачі синтезу звука та музики, розпізнавання мови, обробки відео- та графічної інформації, виконання комунікаційних функцій.

SIMD-обробка потоків даних значно прискорює виконання мультимедійних алгоритмів, для яких є характерне виконання ідентичних операцій над великими масивами однотипних даних – 16-бітними відліками цифрового мовного сигналу, 8-бітні коди кольору пікселя.

Процесор підтримує операцію MAC – множення з накопиченням, у ньому вперше був застосований новий вид арифметики з насиченням: якщо результат операції не вміщується у розрядній сітці, то замість переповнення або втрачання порядку установлюється відповідно максимально або мінімально можливе значення числа.

Pentium MMX має дві окремі кеш-пам'яті – кеш команд та кеш даних, обсяг кожного з них 16 кбайт.

Ефективність за швидкістю виконання мультимедійних додатків у Pentium MMX на 60% вища ніж у процесора Pentium за однакових тактових частот. Регістри MMX Pentium MMX суміщені з 64-розрядними регістрами з плаваючою точкою, що забезпечує сумісність з архітектурою Pentium і дозволяє використовувати раніше розроблене програмне забезпечення. Pentium MMX за 50 тактів може переключатися в режим обчислень з плаваючою точкою і використовуватися як універсальний.

Процесор Pentium Pro (P6) відноситься до шостого покоління; він має кеш другого рівня на 512 кбайт, розміщений у тому ж корпусі, який працює на частоті ядра – 200 МГц. Архітектура процесора спрямована на збільшення паралельно виконуваних віток програми, причому команди можуть виконуватись не в тому порядку, в якому вони розташовані у програмі, але

послідовність запису результатів у пам'ять або виведення їх у порти зберігається відповідно до програми, це так зване динамічне виконання інструкцій. Передбачено спекулятивне виконання команд з випередженням, переупорядкування команд по конвеєрах з метою вирівнювання часу їх виконання, що також підвищує продуктивність процесора. Процесор має різні незалежні шини для з'єднання процесорного ядра з кеш-пам'яттю та основною пам'яттю. Перша шина працює на тактовій частоті процесора, а друга – з частотою системи. Слід зазначити, що 16-розрядні додатки виконуються на Pentium Pro не швидше ніж на Pentium з такою ж тактовою частотою. Це пояснюється частим перезавантаженням виконавчого багатоступінчатого конвеєра з короткими даними.

До МП шостого покоління відносяться також МП Pentium II, Pentium III, Celeron та Xeon. Загальними рисами цих процесорів є те, що ядро процесора вміщує кілька конвеєрів, до яких підключаються виконуючі пристрої для операцій над цілими числами, звернень до пам'яті, передбачення переходів та обчислень з плаваючою точкою. Кілька різних виконавчих пристроїв можуть об'єднуватися на одному конвеєрі.

Pentium II є більш швидкий ніж Pentium Pro з підтримкою MMX, але не пристосований до роботи у багатопроцесорних системах. Pentium III – це подальша розробка Pentium II, його основною відміною від попередніх моделей є розширення SIMD-інструкцій, засноване на новому блоці 128-розрядних регістрів XMM. Блок дозволяє одній інструкції виконувати операції одночасно над чотирма комплектами 32-розрядних операндів з плаваючою точкою. При виконанні повних інструкцій обладнання для виконання традиційних операцій FPU/MMX не використовується, це дозволяє одночасно виконувати інструкції MMX з інструкціями над операндами з плаваючою точкою, а також логічні інструкції.

Сімейство Xeon – Pentium II Xeon та Pentium III Xeon призначені для роботи у складі серверів, які можуть бути об'єднані у системи з FRC-режимом, в якому процесор працює як перевірний з надлишковим контролем функціональності у двопроцесорній, а також у симетричних 1-, 2-, 4- та 8-процесорних системах. Процесори Pentium II Xeon мають частоту шини 100 МГц і частоту ядра 500 МГц, вторинний кеш 1 Мбайт, виконують стандартний набір інструкцій Pentium II та MMX. Процесори Pentium III Xeon мають частоту шини 133 МГц при частоті ядра до 1000 МГц. Вторинний кеш сягає 2 Мбайти.

Процесори Celeron призначені для однопроцесорних конфігурацій, мають частоту ядра до 3,2 МГц, для Celeron D частота шини пам'яті складає 533 МГц, обсяг кеш L2 дорівнює 256 кбайт.

До сьомого покоління процесорів Intel належить Pentium 4. З програмної точки зору він є процесор з архітектурою x86 з розширенням системи команд SIMD – SSE2, який виник в 2001 році. У 2005 році був представлений процесор Pentium 4 з 64-бітним розширенням (EM64T) і додатковими командами SSE3. Цей процесор можна віднести вже до восьмого покоління. Процесор має так звану архітектуру NetBurst і працює на високих частотах ядра до 3,4 ГГц і шини

пам’яті 800 МГц; кеш L2 становить 512 кбайт, L3 – 2048 кбайт. Процесор спрямовано на використання у мультимедійних потокових інтернет-додатках: обробка відеоінформації у реальному часі, задачі кодування-декодування сигналів, IP-телефонія, шифрування даних, телебачення високої чіткості тощо. Pentium 4 побудований за технологією гіперпотоковості (hyperthucading), яка розміщує на одному кришталі два логічних процесори, які одночасно виконують два потоки інструкцій.

Завдяки тому, що два самостійних процесори реалізовані на одному кришталі, можна об’єднати їх кеш та зменшити час передавання даних між ними. Двоядерні процесори дають значне підвищення продуктивності, тільки якщо програмне забезпечення підтримує паралельні обчислення, але не у два рази відносно до одного ядра через те, що потрібний час на перерозподіл навантаження між ядрами, розподіл пам’яті тощо.

Мультиядерні процесори мають кілька функціонально завершених процесорів на одній шині. Кожне ядро має власні кеші L1 та L2 та кеш трас. Розмір кожного кеша L2 сягає 2 Мбайт. Інтерфейс системної шини може бути спільним або роздільним.

Двоядерний процесор Core 2 DUO E6850 має штатну частоту ядра 3 ГГц і частоту шини 1333 МГц. Чотири ядерний процесор Core DUO Q6600 має штатну частоту ядра 2,4 ГГц і частоту шини 1067 МГц. Процесори мають архітектуру Intel® 64®, що дозволяє підтримувати 64-розрядні обчислення, збільшувати обсяг пам’яті, яка адресується, підвищувати рівень безпеки у віртуальних та спеціалізованих обчислювальних середовищах.

Слід зазначити., що всі сучасні процесори у своїй архітектурі мають елементи, які призначені для зменшення енергоспоживання. Вони також не вимагають особливих умов для охолодження.

У табл. 7.3 наведені основні технічні характеристики чотириядерних процесорів Intel® Xeon® серії 5300.

Таблиця 7.3 – Основні технічні характеристики чотириядерних процесорів Intel® Xeon® серії 5300

Процесор	Ємність кеш-пам’яті, Мбайт	Частота системної шини, МГц	Тактова частота, ГГц
Intel® Xeon® X5355	8	1333	2,66
Intel® Xeon® X5345	8	1333	2,33
Intel® Xeon® X5320	8	1066	1,86
Intel® Xeon® X5310	8	1066	1,6
Intel® Xeon® X5335	8	1333	2

Контрольні запитання:

1 Чи можна сказати, що процесори Pentium усіх моделей мають фон-нейманівську архітектуру?

2 Які технічні новації притаманні процесору Pentium?

3 Яку структуру має кеш-пам’ять процесора Pentium?

4 Скільки розрядів має шина даних процесора Pentium?

5 Які засоби енергозбереження має процесор Pentium?

6 Які загальні риси мають процесори шостого покоління Pentium II, Pentium III, Celeron та Xeon?

Контрольні запитання підвищеної складності:

1 В яких процесорах використовуються архітектури багатопроцесорних систем SIMD, MISD, MIMD?

2 Поясніть, які особливості має архітектура двоядерних процесорів?

3 Які особливості має архітектура мультиядерних процесорів?

7.3.3 Процесори провідних фірм AMD

Вхідний контроль:

1 Який з відомих Вам процесорів фірми Intel має найвищу тактову частоту і яка вона є?

2 В яких областях використовують мультиядерні процесори?

Процесор Athlon(K7) фірми AMD є високопродуктивним за рахунок високої тактової частоти (з 1 ГГц), а також особливої суперконвеєрної, суперскалярної архітектури. Процесор має триканальний декодер інструкцій x86, який вибирає їх з пам’яті через кеш, та потужний блок передбачення розгалужень. Інструкції мови Асемблер 86, які можуть мати довжину від 1 до 15 байт, перетворюються в уніфіковані макрооперації. За кожний такт блок керування інструкціями може отримати до трьох інструкцій від декодера. Сам блок є буфер, який перевпорядковує інструкції та розподілює макрооперації по виконавчих пристроях процесора, перейменовує регістри та приймає результати обчислень з виконавчих конвеєрів. Процесор має три незалежних конвеєри цілочислових обчислень, три конвеєри для обчислення адрес операндів та триканальний пристрій для обчислень з плаваючою точкою. У процесорі вперше застосована повністю конвеєризована суперскалярна архітектура зі зміненням порядку виконання інструкцій для обчислень з плаваючою точкою. Процесор виконує усі інструкції традиційного FPU(x87), MMX та 3D Now!. Продуктивність процесора сягає 4 Гфлопс при обчисленнях з одинарною точністю.

Система команд вміщує стандартний набір інструкцій Intel шостого покоління, MMX і розширену технологію 3D Now!. 12 нових цілочислових SIMD-інструкцій призначені для виконання операцій, пов’язаних з розпізнаванням мови, відеокодуванням, 7 інструкцій пов’язані з прискоренням передавання даних, 5 інструкцій реалізують функції сигнальних процесорів, вони дозволяють підвищувати продуктивність таких додатків, як програмні моделі (ADLS), процесори об’ємного звучання тощо.

Первинний кеш процесора становить 128 кбайт (64 для даних і 64 для команд) і високопродуктивний інтерфейс розрядністю даних 72 біти для

вторинного кеш обсягом 8 Мбайт. Процесори Athlon можуть працювати у багатопроцесорних системах.

Для підтримки процесора Athlon фірма AMD випустила чипсет, який складається з двох мікросхем: системного контролера та периферійного контролера. Системний контролер забезпечує зв'язок процесора, динамічної пам'яті, порту AGP та шини персонального комп'ютера PCI. Чипсет допускає установлення до трьох модулів DIMM SDRAM, з сумарним обсягом 768 Мбайт. Шина PCI може обслуговувати до 6 контролерів. Системний контролер має буфери, які забезпечують можливість одночасного виконання обміну даними між парами "користувачів" (процесор, пам'ять, порт AGP і шина PCI). Периферійний контролер має міст шин PCI-ISA, який підключає усі системні пристрої PC-сумісного комп'ютера (контролери переривань, прямого доступу до пам'яті, клавіатури та миші, таймер, інтерфейс флеш-BIOS). Контролер USB виконує функції центрального концентратора і має 4 порти для підключення зовнішніх пристроїв.

Процесор Athlon-64 та Opteron мають власний контролер пам'яті з окремою шиною, до якої безпосередньо підключаються модулі пам'яті. У Athlon-64 шина даних пам'яті є 64 бітна, а у Opteron – 128-бітна. Завдяки цьому затримка доступу до пам'яті становить 45 нс. Процесори Opteron призначені для серверів (1–8-процесорних систем) та робочих станцій (1–4-процесорних систем) мають три 16-бітних інтерфейси Hyper-Transport з пропускною здатністю 19,2 Гбайт/с. Кожний процесор входить у багатопроцесорну систему і має також доступ до пам'яті інших процесорів.

Якщо надати кожному процесору свій адресний простір, можна реалізувати багатокомп'ютерну систему, в якій за сумісної паралельної роботи вони можуть обмінюватись лише повідомленнями через інтерфейс Hyper-Transport.

Фірма AMD випускає двоядерні процесори Athlon-64x2 для робочих станцій, а також процесори Opteron (64-бітний) і Athlon MP (32-бітний) для мультипроцесорних серверів і робочих станцій.

Для настільних комп'ютерів призначені 64-бітні процесори Athlon 64 FX і Athlon 64, а також 32-бітні Semptron і Athlon XP.

Контрольні запитання:

1 Які особливості має архітектура процесорів фірми AMD?

2 Який з відомих Вам процесорів фірми AMD має найвищу продуктивність?

Контрольні запитання підвищеної складності:

1 Що таке додатки MMX?

2 Що таке технологія 3D Now!?

3 Чи підвищує сьогодні якість зображень у іграшках застосування у ПК мультиядерних процесорів?

7.3.4 Продуктивність мікропроцесорів та її оцінювання

Вхідний контроль:

1 Як Ви пов'язуєте продуктивність МП з тактовою частотою?

2 З якою метою Ви обираєте свій домашній ПК з максимальною продуктивністю?

Технічна пікова продуктивність МП – це теоретичний максимум швидкодії комп'ютера або МПС за ідеальних умов. Вона визначається як кількість обчислювальних операцій, які виконуються за секунду усіма арифметико-логічними пристроями, які є у процесорі. Максимальна швидкодія досягається при обробленні нескінченної послідовності не зв'язаних між собою за даними команд, які також не конфліктують при доступі до пам'яті. Практично жодна система не може довго працювати з максимальною продуктивністю, але майже всі комп'ютери досягають 0,8…0,95 максимуму.

Для оцінки пікової продуктивності потрібно знати тактову частоту процесора, розрядність оброблюваних даних, пропускну здатність та кількість внутрішніх шин, перелік функціональних пристроїв.

Відповідність між одиницями вимірювання тактової частоти та продуктивності процесора установлюється для одного конвеєра такою: 1 МГц відповідає 1 MFLOPS або 1 MIPS пікової залежно від типу операції – з плаваючою або фіксованою точками. Для суперскалярних процесорів пікова продуктивність обчислюється множенням значення тактової частоти на кількість паралельно виконуваних операцій.

Складна архітектура сучасних високопродуктивних процесорів – суперскалярна та суперконвеєрне оброблення даних, багаторівнева пам'ять тощо призводять до того, що характеристики швидкодії на рівні внутрішніх пристроїв істотно залежать від програми та даних.

У світовій практиці набуло широкого розповсюдження використання наборів задач (тестів), характерних для визначеної області застосування обчислювальної техніки, для оцінки продуктивності комп'ютерів. Час, необхідний для вирішення кожної задачі з набору, складає основу для обчислення індексу продуктивності, який є відносною оцінкою.

Першу групу тестів складають компанії-виробники для попередньої оцінки. Головна їх особливість полягає в тому, що вони орієнтовані на порівняння обмеженої кількості комп'ютерів, які часто відносяться до одного сімейства. Прикладом такої оцінки для МП з архітектурою x86 компанія Intel запропонувала індекс продуктивності iCOMP (Intel Corporative Microprocessor Performance), а в якості еталонного прийнятий процесор I80486 SX-25, для якого індекс дорівнює 100. Індекс iCOMP визначається при виконанні суміші операцій, яка складається з 67% операцій над 16-розрядними цілими, 3% операцій над 16-розрядними числами з плаваючою точкою, 25% – над 32-розрядними цілими та 5% – над 32-розрядними числами з плаваючою точкою і оцінює тільки продуктивність мікропроцесора, а не системи.

Стандартні тести, орієнтовані на порівняння широкого спектра комп'ютерів, складаються незалежними експертами або групами, об'єднуючими потужних виробників комп'ютерів. Це виключає орієнтацію тестів на конкретного виробника. Так, для оцінки серверів, які оброблюють трансакції у реальному часі, використовується набір тестів TPP–C (Transaction Processing Performance Council), а продуктивність оцінюється кількістю трансакцій, які виконуються за хвилину.

Існують також тести, які орієнтовані на вибір комп'ютерів та програмного забезпечення, найбільш придатних для вирішення певного кола задач. Такі тести дають найбільш точні оцінки продуктивності для конкретного класу додатків.

Найбільшої популярності набули пакети тестових програм компанії SPEC (Standard Performance Evalution Corporation) – SPEC 89, SPEC 92 тощо. Задачі, які входять у тестові пакети, вражають своєю різноманітністю і торкаються широких областей наукової та прикладної діяльності: задача з теорії мереж, інтерпретатор мови Lisp, Unix – утиліта пакування тестового файла 1 Мбайт, який стискається у 20 разів, моделювання керування руху, робота з використанням відеосистеми, моделювання вуха людини тощо. В останні версії тестів були включені задачі для оцінки продуктивності процесора у багатозадачному режимі для однорідного навантаження (комп'ютер виконував багато копій одної програми), а результатом вимірювань був нормований загальний час виконання всіх копій.

Тестовий пакет SYS Mark 2007 вимірює продуктивність процесорів у складі ПК у реальних додатках. Тест вміщує сценарій, в якому моделюється підготовка веб-сайта, який навчає. Медіаконтент використовує Adobe Illustrator CS2, Adobe Photoshop CS2, Macromedia Flash та Microsoft Power Point 2003. За сценарієм виготовляється відеоролик з використанням нелінійного монтажу та різних ефектів. Ролики монтуються з різних джерел і вміщують статичні зображення.

Тестовий пакет SYS Mark 2007, Productivity модулює типову офісну роботу: використання e-mail, обробку даних, керування проектом і роботу з документами. Тест працює з додатками: Microsoft Excel2003, Microsoft Outlook 2003, Microsoft Power Point 2003, Microsoft Word 2003, Microsoft Project 2003 та WinZip 10.0.

Тестовий пакет SYS Mark 2007, 3D присвячено створенню архітектурної презентації, яка вміщує фотореалістичне зображення об'єкта і ролик, де проектований будинок облітається літаком. У сценарії використовуються додатки AutoDesk 3ds Max 8 та Sketch Up 5.

Контрольні запитання:

1 Що таке технічна продуктивність МП?
2 Як вона оцінюється?
3 Що таке реальна продуктивність МП?
4 Які Ви знаєте способи оцінки реальної продуктивності?
5 Які шляхи підвищення продуктивності МП Ви знаєте?

6 Які задачі входять до пакетів тестових програм оцінювання продуктивності МП?

Контрольні запитання підвищеної складності:

1 Чому у пакети тестових програм залучають задачі оброблення зображень?

2 Які ще задачі Ви могли б запропонувати для використання у пакетах тестових програм?

8 ВИКОРИСТАННЯ СУЧАСНИХ МІКРОПРОЦЕСОРІВ У ТЕЛЕКОМУНІКАЦІЙНОМУ ОБЛАДНАННІ (Для поглибленого вивчення)

Вхідний контроль:

1 Які системи телекомунікацій Ви знаєте?

2 Які з них працюють на основі мікропроцесорів та мікропроцесорних систем?

Більш ніж 50 % сучасного мережного обладнання використовують персональні комп'ютери на основі процесорів фірми Intel та сумісних з ними або вбудовують їх у пристрої шлюзів, маршрутизаторів, інтегрованих платформ, систем комутації.

В усіх засобах зв'язку, будь-то квазіелектронні АТС, електронні АТС, цифрові системи комутації, шлюзи, маршрутизатори та інші мережні пристрої, інтегровані платформи тощо, використовуються багатопроцесорні системи як менш двопроцесорні, або такі, що складаються з двох комп'ютерів. Можливе розподілення між кількома різними процесорами в одній системі функцій керування та цифрового оброблення сигналів. Тоді це універсальний процесор та до 20 – 30 процесорів цифрового оброблення сигналів, як правило, фірм-виробників Motorola, Analog Devices, Texas Instruments.

Нижче наведені приклади застосування МП фірми Intel у телекомунікаційному обладнанні.

Централізовані керувальні системи складаються з центрального пристрою керування, периферійних пристроїв керування. Керування усіма процесами на вузлі комутації квазіелектронних АТС виконує двомашинні керувальні комплекси. Наприклад, система комутації "Квант-Е" використовує два персональних комп'ютери на МП Intel 486DX, один комп'ютер виконує усі функції керування в автоматичному режимі, а другий знаходиться у гарячому резерві. У синхронному режимі обидва комп'ютери синхронно виконують одні й ті ж самі функції і на певних станах оброблення порівнюють результати з метою підвищення надійності та достовірності оброблення викликів. У режимі розподілу навантаження обидва комп'ютери обслуговують виклики паралельно, що підвищує продуктивність приблизно в два рази.

Система комутації "Квант-Е" використовує двомашинну систему на базі системних плат IBM PC "PCA-6143 P" із процесором Intel 486DX з тактовою частотою не менше ніж 133 МГц, а останні моделі використовують процесори Pentium.

Багато фірм, які випускають шлюзи, використовують як керувальний та оброблюючий пристрій персональні комп'ютери на мікропроцесорах фірми Intel та Intel-сумісних. Так, фірма Clarent використовує ПК з двома процесорами Pentium Pro для оброблення, передавання та приймання телефонних дзвінків та факсів по мережі TCP/IP.

Фірма Comdial випускає шлюз IP-телефонії на базі кількох серверних пристроїв на 2-х або 4-х Pentium Pro, а кодування/декодування мовного сигналу здійснюється на DSP.

Шлюзи фірми Franclin Telecom використовують для кожного каналу IP-телефонії один кодек G.729 для стиснення мовних сигналів у 8 разів на DSP. У цілому шлюзи працюють під ОС UNIX на універсальному процесорі і мають 24 голосових канали.

WebPhone Exchange (WGX) Server компанії Netspeak переводить дзвінки зі звичайних телефонів у мережу IP та дзвінки з додатка Webphone у телефонну мережу. WGX має процесор Pentium (200 МГц) та працює з ОС NT. Дзвінок за допомогою WGX може бути спрямований з WebPhone на звичайний телефон і навпаки, з телефону на телефон, зі сторінки Web на звичайний телефон. Це потребує не менше двох серверів WGX, які підтримують кодек Microsoft GSM. WGX мають доступ до інформації про маршрути WGX через Connection Server. Система здатна реєструвати події у реальному часі завдяки серверу NetSpeak Database Services (DBS) і підтримувати управління викликами у реальному часі.

Internet Gateway компанії Rockwell дозволяє спілкуватися користувачам Web і Internet та співробітникам центру обслуговування без телефону або додаткової лінії при збереженні з'єднання у вузлах Web. Сервер Internet Gateway поставляється разом з додатком WebPhone і має апаратну платформу мультимедійного ПК на базі ОС Windows.

Фірма VocalTec випускає шлюзи IP-телефонії – VocalTec Telephony Gateway Release 3.1. Ця система на базі NT для організації моста між телефонною мережею та Intranet, Internet з підтримкою дзвінків з телефону на ПК та з Web на телефон. Після з'єднання зі шлюзом автоматичний секретар запитує в абонента телефонний номер адресата, після чого номер вводиться з клавіатури. Місцевий шлюз автоматично визначає, що виклик треба адресувати віддаленому шлюзу. Система надає також послугу інтерактивній голосовій відповіді, відправки факсів у реальному часі або з проміжним зберіганням. Шлюз використовує механізм автоматичного визначення для запобігання роз'єднання в результаті довгих періодів мовчання. Модуль Surf&Call для броузера Web дозволяє дзвонити з сервера Web на звичайний телефон. Користувачі можуть звертатися до голосової пошти за технологією DTMF, а віддалені користувачі ПК отримують зручний доступ до мережі через VocalTec Internet Phone. Ця технологія буде корисною корпораціям з кількома офісами, якщо вони надають додаткові платні послуги провайдерам Internet. Версія шлюзу 3.2 замінює магістральні лінії (одноступеневий набір номера). Шлюз забезпечує 8 ліній.

Шлюз Інтернет-телефонії фірми Intertel виконано у вигляді автономного сервера з окремими платами обробки голосового сигналу за протоколами H.323 та G.723.1. Він підтримує сторінки Web з кнопками з функцією „натисни, щоб поговорити". Система побудована на ПК, який працює під ОС NT, і вимагає процесора, який підтримує цю ОС, тобто не нижче Pentium.

Обладнання IP-телефонії фірми Cisco має архітектуру AVVID (Architecture for Voice, Video and Integer Data) і пропонує широку номенклатуру

обладнання для IP-телефонії, яке призначене для застосування у корпоративних мережах, створення серверів багатофункціональних систем цифрової телефонії, підключення її до мереж загального користування, надання сучасних сервісів для абонентів. Усі пропоновані системи можуть масштабуватись від кількох десятків до сотень тисяч абонентів, у тому числі віддалених один від одного географічно.

До базових пропозицій компанії можна віднести такі: IP-телефони з власною IP-адресою, які можна підключати до будь-якого H.323-сумісного пристрою та користуватись типовим ПЗ, наприклад, Microsoft NetMeeting.

Сервер Call Manager реалізує керування телефонними з'єднаннями, сервісами в системі адміністрування тощо. Сервери можуть об'єднуватись у кластери.

ПЗ Call Manager установлюється на ПК не нижче Pentium під Windows NT, підключений до IP-мережі, IP-телефон або програмно реалізований віртуальний телефон, і забезпечує такі можливості, як утримання дзвінка, переспрямування дзвінка, перехоплення дзвінка, ідентифікацію абонента, який викликає. Інтерфейс SMDI ПЗ Call Manager забезпечує інтеграцію з голосовою поштою, інтерактивними системами мовного зв'язку для складання звітності про телефонні послуги, ПЗ для білінга.

За обсягом продаж обладнання IP PBX на світовому ринку лідирує компанія CISCO. Вона проводить політику переходу до єдиної системи комунікацій за протоколом IP. ПЗ CISCO Call Manager реалізують функції телефонної офісної УАТС і працюють на відокремленому сервері під Windows 2000 з процесором не нижче Pentium III Xeon. Система може бути розподіленою або централізованою, масштабується, дозволяє швидко нарощувати мережу для передавання голосу, підмикати нові офіси, підтримувати систему уніфікованої обробки повідомлень Unity (інтегровану з Microsoft Exchange та Lotus Notes) тощо. Версія Call Manager 3.3.2 здатна обслуговувати до 30 тис. IP-телефонів, має розширену сигналізацію Q.SIG (протокол SIP поки не підтримується). Call Manager може використовуватись у поєднанні з традиційною УАТС або як окреме рішення. Кілька серверів під Call Manager можуть об'єднуватись у кластери (до 8 вузлів), сервер може знаходитись у клієнта, в оператора, здійснювати все керування IP-телефонією корпорації, адміністрування та налаштування мережі. Нова модульна IP PBX ICS 7750 з функціями підтримки адміністратора викликів, функцій голосової пошти та IVR. На основі архітектури AVVID CISCO пропонує кілька десятків продуктів. Завдяки ОС IOS CISCO маршрутизатори можуть виконувати функції малих АТС та підтримують сотні IP-телефонів, адаптери ATA186, призначені для підключення до Ethernet аналогових телефонів, модемів, факсів.

Фірма Siemens представляє IP BPX HiPath серії 3000 та 5000 на ОС Windows NT, процесори не нижче Pentium на 250-300 користувачів та УАТС Hico з підтримкою IP.

Комунікаційна платформа 3300 ICP канадської компанії MITEL NETWORKS Ltd, основана на технології IP, призначена для середнього та корпоративного бізнесу з приватними офісами. Усі компоненти платформи

керуються через Web-браузери. Одна платформа 3300 ICP підтримує 100 IP-телефонних номерів, забезпечує голосовий зв'язок, сигналізацію, централізоване оброблення даних та системні телекомунікаційні ресурси. Контролер підключається до клієнтської локальної мережі та до IP-пристроїв за допомогою чотирьох портів комутатора 10/100 Base T Ethernet. Універсальний блок аналогових послуг 3300 Mitel Network вміщує 16 внутрішніх аналогових телефонних ліній та 4 зовнішніх міських телефонних лінії. Універсальний блок мережних послуг 3300 Mitel Network забезпечує T1 або E1 з'єднання і підтримує до двох інтерфейсів T1 або E1 на модуль. Протоколи, підтримувані T1 інтерфейсам: T1CAS – Digital E TAM, Digital CO, Digital DID T1 CCS. Первинна швидкість ISDN (4ESS, 5ESS, DHS 100, DSM 250, N 12, N 13), XNET через PRI, DASSII, MSDN/DPNSS. Протоколи, підтримувані E1 інтерфейсом: Q.Sig, ISDN, XNET через PRI, DASSII, MSDN/DPNSS. Блок мережних послуг R2NSU забезпечує з'єднання з міжстанційними трактами R2, використовуючими цифрову сигналізацію MF-R2. Один блок R2 NSU підтримує до двох трактів, додаткові тракти можуть додаватись до 3300 ICP при з'єднанні двох NSU. Блок BRI NSU забезпечує з'єднання для передачі голосу та даних через інтерфейси основного доступу (BRI) для ISDN. Один блок підтримує 15 BRI U-інтерфейсів. IP-консоль 5550 основана на базі ПК має високоякісний графічний інтерфейс (GUI). Вимоги до ПК, на якому інсталюється консоль, такі: процесор Pentium, ОС Microsoft Windows 2000 Professional, 128 MB доступної RAM 4GB жорсткого диску тощо.

АТС системи Квант-Е використовує як центральний пристрій управління двомашинний комплекс на базі системних плат з процесором Intel 386DX-40 (або сумісних). Фірма Micom (відділення Nortel) розробила плату Vo/IP Phone/Fax IP Gateway – шлюз IP-телефонії. Базовими блоками шлюзу є голосові плати та комплект програмного забезпечення для конфігурації системи, керування викликами, налагодження, протоколів пріоритезації для маршрутизаторів тощо. Micom розробила ефективні методи подавлення пауз, завдяки яким пропускна здатність знижується до 7 кбіт/с для однієї розмови за рахунок зупину передавання, коли абонент робить паузи між словами. Досить розвинене програмне забезпечення Micom потребує дуже малих процесорних ресурсів: йому достатньо процесора I80486.

При виборі типу мікропроцесора або мікроконтролера для мережного пристрою певного призначення визначальним є наявність апаратної підтримки комунікаційних функцій та висока продуктивність. За рахунок наявності вбудованих функцій необхідна частота процесора може бути знижена. Для обробки пакетів у сучасних мережах важлива наявність у процесорі кешу 1- та 2-го рівнів, що забезпечить високу швидкість мережі.

Фірми AMD та SUN випускають сервери Sun Fire для центрів обробки даних на основі МП AMD Opteron, які є 64-розрядними. У табл. 8.1 наведені основні характеристики таких серверів.

У табл. 8.2 та 8.3 наводяться характеристики робочих станцій та серверних систем Sun Fire.

Таблиця 8.1 – Сервери Sun Fire для центрів обробки даних

	Sun Fire X2100	Sun Fire X2200	Sun Fire X4100	Sun Fire X4200	Sun Fire X4500	Sun Fire X4600	Sun Blade X8400	Sun Blade X8420
Проце-сори AMD Opteron	1 одноядерний або двоядерний (серії 100 або 1000)	До 2-х двоядерних	До 2-х одноядерних або двоядерних	До 2-х одноядерних або двоядерних	До 2-х двоядерних	До 8-ми одноядерних або двоядерних	До 4-х двоядерних	4 двоядерних (серії 8000)
Тактова частота	До 3,0 ГГц	До 2,6 ГГц	До 3,0 ГГц	До 3,0 ГГц	До 3,0 ГГц	До 3,0 ГГц	До 2,6 ГГц	2,4 ГГц, 2,6 ГГц, 2,8 ГГц
Кеш-пам’ять 2 рівня	По 1 МБ на ядро	По 1 МБ на ядро	По 1 МБ на ядро	По 1 МБ на ядро	По 1 МБ на ядро	По 1 МБ на ядро	По 1 МБ на ядро	1 МБ на ядро, кеш L2
Опера-тивна пам’ять	До 8 ГБ	До 64 ГБ	До 16 ГБ	До 16 ГБ	До 16 ГБ	До 64 ГБ	До 64 ГБ	До 64 ГБ
Внутріш-ні диски	До 2-х дисків	До 2-х дисків	До 2-х дисків з DVD-ROM або до 4-х дисків без DVD-ROM	До 4-х дисків з DVD-ROM	До 48-ми дисків	До 4-х дисків	До 2-х дисків	До 2-х дисків

Продовження табл. 8.1

	Sun Fire X2100	Sun Fire X2200	Sun Fire X4100	Sun Fire X4200	Sun Fire X4500	Sun Fire X4600	Sun Blade X8400	Sun Blade X8420
Типи внутріш-ніх дисків	80/250/500 ГБ 7200 об/хв, SATA	250/500 ГБ 7200 об/хв, SATA	36/73 ГБ 10000 об/хв, SAS	36/73 ГБ 10000 об/хв, SAS	250/500 ГБ 7200 об/хв, SATA, макс. 24 ТБ	10000 об/хв, SAS	73 ГБ 10000 об/хв., SAS; 80 ГБ 5400 об/хв, SATA	73 ГБ/146 ГБ SAS або 80 ГБ SATA, “гаряча заміна”
З’єдну-вачі введення-виведення	До 2-х слотів PCIe x8	До 2-х слотів PCIe x8; 2 слоти PCI-X	2 слоти PCI-X	9 слотів PCI-X	2 64-розряд-них слоти PCI-X	8 слотів PCI-X; 4 слоти x8 PCIe; 2 64-розряд-них слоти PCI-X 100 МГц	6 інтерфейсів PCI-Express	6 інтерфейсів PCI-Express
Віддалене керування	Стандартний вбудований сервісний процесор з підтримкою IPMI 2,0	Вбудований Sun Integrated Light Out з повною підтримкою KVMS	Вбудований Sun Integrated Light Out з повною підтримкою KVMS	Вбудований Sun Integrated Light Out з повною підтримкою KVMS	Sun Integrated Lights Out Manager, N1 System Manager	Вбудований сервісний процесор Sun Integrated Lights Out Manager з повною підтримкою KVMS	Sun Integrated Lights Out Manager з повною підтримкою KVMS	Sun Integrated Lights Out Manager в стандартній конфігурації

Закінчення табл. 8.1

	Sun Fire X2100	Sun Fire X2200	Sun Fire X4100	Sun Fire X4200	Sun Fire X4500	Sun Fire X4600	Sun Blade X8400	Sun Blade X8420
Операційна система	Solaris 10, Red Hat Enterprise Linux, SUSE Linux Enterprise Server та Microsoft Windows	Solaris 10, Red Hat Enterprise Linux, SUSE Linux Enterprise Server, Microsoft Windows та VMware ESX Server	Solaris 10, Red Hat Enterprise Linux, SUSE Linux Enterprise Server, Microsoft Windows	Solaris 10, Red Hat Enterprise Linux, SUSE Linux Enterprise Server, Microsoft Windows та VMware ESX Server	Solaris 10, Red Hat Enterprise Linux, SUSE Linux Enterprise Server	Solaris 10, Red Hat Enterprise Linux, SUSE Linux Enterprise Server, Microsoft Windows та VMware GSX та Virtual Server	Solaris 10, Red Hat Enterprise Linux, SUSE Linux Enterprise Server та Microsoft Windows	OS Solaris 10, Red Hat Enterprise Linux 4, SUSE Linux Enterprise Server 9, Windows Server 2003 EE та SE VMware ESX Server 3.0.1

Таблиця 8.2 – Робочі станції Sun Fire

	Робочі станції Sun Ultra 20	Робоча станція Sun Ultra 40
Процесори AMD Opteron	1 одноядерний або двоядерний (серії 100 або 1000)	До 2-х одноядерних або двоядерних
Тактова частота	До 3,0 ГГц	До 3,0 ГГц
Кеш-пам’ять 2-го рівня	По 1 МБ на ядро	По 1 МБ на ядро
Оперативна пам’ять	До 8 ГБ	До 8 ГБ
Внутрішні диски	До 2-х дисків	До 4-х дисків
Типи внутрішніх дисків	80/250/500 ГБ, макс. 1 ТБ, SATA	80/250/500 ГБ, макс. 2 ТБ, SATA
З’єднувачі введення-виведення	До 2-х слотів PCIe x16; 2 слоти PCIe x1; до 4-х слотів PCI	2 слоти PCI-X x16; 2 слоти PCIe x4; 2 стандартних слоти PCI
Графічна система	PCI: ATI RageXL, ATI ES 1000; PCIe: NVIDIA Quadro NVS 285; FX 540, FX 560, FX 1400, FX 1500, FX 3450, FX 3500	PCIe: NVIDIA Quadro NVS 285 DDR2, FX 560, PCIe/SLI-Ready: NVIDIA Quadro FX 1500, FX 3500, FX 5500
Програмне забезпечення	Попередньо установлені Sun Studio, Sun Java Studio Enterprise, Sun Java Studio Creator та NetBeans IDE	Має безкоштовну ліцензію на Sun N1 Grid Engine 6. Попередньо установлені Sun Studio, Sun Java Studio Enterprise, Sun Java Studio Creator та NetBeans IDE
Операційна система	Solaris 10, Red Hat Enterprise Linux, SUSE Linux, Microsoft Windows та Microsoft Windows Vista	Solaris 10, Red Hat Enterprise Linux, SUSE Linux, Microsoft Windows

Таблиця 8.3 – Серверні системи Sun Fire

	Модульна система Sun Blade 8000	Модульна система Sun Blade 8000 P
Серверні модулі	10 серверних модулів Sun Blade 84x0 в один корпус	10 серверних модулів Sun Blade 84x0 на шасі
Розміри	Висота 10 RU, 2 модуля PCIe Express Module, до 20 з'єднувачів на шасі. До 4-х модулів PCIe Network Express на шасі	14 U, 2 модуля Network Express стандарту PCIe, загальною ємністю 40 портів введення-виведення на шасі або 120 портів введення-виведення на стійку
Віддалене керування	Модуль моніторингу корпусу (Chassis Monitoring Module, CMM) забезпечує можливості моніторингу корпусу та установлених у ньому компонентів; є можливість установлення модуля з надмірністю (подвійний CMM) для моніторингу з високою доступністю	Кожен сервер має вбудований сервісний процесор ILOM з підтримкою: DMTF CLI через SSH, графічний інтерфейс користувача на основі браузера по HTTPS/HTTP, IPMI 2.0, SNMP v1, v2 та v3, віддалений KVM по Ethernet, віддалені носії інформації по Ethernet, сумісні з ПЗ керування Sun N1 System Manager
Інтерфейси	Інтерфейс DB9 RS-232C (включаючи електронний ключ DB9 на RJ45), два порти Gigabit Ethernet для віддаленого керування	Інтерфейс DB9 RS-232C (включаючи електронний ключ DB9 на RJ45), два порти Gigabit Ethernet для віддаленого керування
Підтримувані протоколи	SSH, SNMP v1, v2 та v3, IPMI, HTTPS, віддалена клавіатура, відео, миша та система зберігання даних (RKVMS), інтерфейс командного рядка DMTF/SMASH для керування по мережі або з використанням послідовного порту	SSH, SNMP v1, v2 та v3, IPMI, HTTPS, віддалена клавіатура, відео, миша та система зберігання даних (RKVMS), інтерфейс командного рядка DMTF/SMASH для керування по мережі або з використанням послідовного порту

Контрольні запитання:

1 Які архітектурні особливості сучасних 32- та 64-розрядних мікропроцесорів фірми Intel зумовили їх широке застосування у телекомунікаціях?

2 З якою метою у шлюзах використовують МП фірми Intel?

3 Під якою операційною системою працюють шлюзи, які використовують МП фірми Intel?

4 Як використовуються ПК на МП фірми Intel у АТС системи Квант-Е?

5 Як використовує фірма CISCO МП Pentium III Xeon?

6 Як використовуються МП фірми Intel в обладнанні IP-телефонії?

Контрольні запитання підвищеної складності:

1 В якому обладнанні використовуються двоядерні процесори Opteron фірми AMD?

2 Під якими операційними системами працюють робочі станції Sun Fire?

3 Які вимоги пред’являють до операційних систем, під якими працюють мультиядерні процесори?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ ДО 1-го МОДУЛЯ

1 Брэй Б. Микропроцессоры Intel: 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro Processor, Pentium II, Pentium III, Pentium 4. Архитектура, программирование и интерфейсы. – 6-е изд./ Пер. с англ. – С.Пб.: БХВ – Петербург, 2005. – 1328 с.: ил.

2 Вычислительные системы, сети и телекомуникации / В. Л. Бройдо – С.Пб.: Питер, 2002. – 688 с.: ил.

3 Мікропроцесорна техніка: Підручник – Ю. І. Якименко. Т. О. Терещенко, Є. І. Сокол, В. Я. Жуйков, Ю. С. Петергеря; За ред. Т. О. Терещенко. – 2-е вид. пер. і доп. – ІВЦ “Вид–во “Політехніка”; “Кондор”, 2004. – 440 с.: іл.

4 Схемотехника электронных систем. Микропроцессоры и микроконтроллеры / В. И. Бойко, А. Н. Гуржий, В. Я. Жуйков, А. А. Зори, В. М. Спивак, Т. А. Терещенко, Ю. С. Петергеря. – С.Пб.: БХВ-Петербург, 2004. – 464 с.: ил.

5 Корнеев В. В., Киселев А. В. Современные микропроцесоры. – М.: НОЛИДЖ, 1998. – 240 с.: ил.

6 Корнеев В. В., Киселев А. В. Современные микропроцесоры. – М.: НОЛИДЖ, 2000. – 320 с.: ил.

7 Ричард Григонис. Шлюзы IP-телефонии // LAN – 1998. № 07–08. (www.osp.ru).

8 Руководство по архитектуре IBM PC AT / Ж. К. Голенкова, А. В. Заблоцкий, М. Л. Марсахин и др.; Под общ. ред. М. Л. Марсахина. – Минск.: ООО “Консул”, 1992. – 949 с.: ил.

9 Каган Б. М. Электронные и вычислительные машины и системы: Учеб. пособие для вузов. – 3-е изд., пер. и доп. – М.: Энергоатомиздат, 1991. – 592 с.: ил.

10 Цифровая и вычислительная техника: Учебник для вузов / Э. В. Евреинов, Ю. Т. Бутыльский, И. А. Мамзелев и др.; Под ред. Э. В. Евреинова. – М.: Радио и связь. 1991. – 464 с.: ил.

11 Проектирование импульсных и цифровых радиотехнических систем: Учеб. пособие для радиотехн. спец. вузов / Ю. П. Гришин, Ю. М. Казаринов, В. М. Катипов и др.; Под ред. Ю. М. Казаринова. – М: Высшая школа, 1985. – 340 с.: ил.

12 Брамм П., Брамм Д. Микропроцессор 80386 и его программирование / Пер. с англ. – М.: Мир, 1990. – 448 с.: ил.

13 Калабеков Б. А. Микропроцессоры и их применение в системах передачи и обработки сигналов: Учеб. пособие для вузов. – М.: Радио и связь, 1988. – 368 с.: ил.

14 Литовкин В. Ф., Митрофанов Ю. Н., Номоконов В. Н. Микропроцесорные системы с фиксированной разрядностью: Учеб. пособие. – Одеса: ОЭИС им. А.С. Попова, 1988. – 72 с.: ил.

15 Литовкин В. Ф., Митрофанов Ю. Н. Цифровая и вычислительная техника: Учеб. пособие. – Одеса, 1989. – 65 с.: ил.
16 Трой Д. Программирование на языке Си для персонального компьютера IBM PC / Пер. с англ. – М.: Радио и связь, 1991. – 432 с.: ил.
17 Валуйский В. Н., Каневский Ю. С., Пиневич М. М., Романкевич А. М., Самофалов К. Г. Прикладная теория прикладных автоматов. – К.: Вища школа, 1987. – 375 с.: ил.
18 Мещерякова Э. В., Митрофанов Ю. Н. Комплексное задание и методические указания к его выполнению по дисциплине «Цифровая и вычислительная техника» для студентов 3-го курса специальности 2305, 2306 и 2307 (часть 1). – Одеса, 1990. – 27 с.: ил.
19 Лебедев О. Н. Микросхемы памяти и их применение. – М.: Радио и связь, 1990. – 234 с.: ил.
20 Потемкин И. С. Функциональные узлы цифровой автоматики. – М.: Энергоатомиздат, 1988. – 320 с.: ил.
21 Гук М. Ю. Аппаратные средства IBM PC. Энциклопедия. 3-е изд. – С.Пб.: Питер, 2008. – 1072 с.: ил.
22 http://www.virt.ru/news/news.htm. «Характеристики микропроцессоров».
23 http://www.infocity.kiev.ua . Курмаз М. «Pentium 4: результаты тестов».
24 http://www.rim2000.com . Толопунский С. «Обзор AMD Duron с ядром Morgan».
25 http://www.rim2000.com . «Процессор Pentium® II Xeon™».
26 ttp://www.infocity.kiev.ua . Холмогорок В. «Процессоры – краткий обзор».
27 http://www.infocity.kiev.ua . Самотин С. «Новые процессоры для мобильных компьютеров».
28 http://www.infocity.kiev.ua . Джок А. «Многопроцессорная обработка на кристалле».
29 http://www.infocity.kiev.ua . Дереза Д. «Pentium 3 - Pentium 4. Новый ход Intel».

2 МОДУЛЬ
9 ПРОГРАМУВАННЯ МІКРОПРОЦЕСОРІВ ФІРМИ INTEL

9.1 Сегментування пам’яті мікропроцесорами

Вхідний контроль:

1 Чи використовують сегментування пам’яті 8-розрядні МП фірми Intel?
2 Поясніть, що є адреса комірки пам’яті?
3 Поясніть, що є вміст комірки пам’яті?

Мікропроцесори фірми Intel призначені для роботи, перш за все, у складі комп’ютерів, які працюють у мультизадачному режимі, та у складі багатопроцесорних систем, де усі процесори можуть звертатися до спільного пристрою пам’яті. Найбільш незахищеною підсистемою МПС у такому застосуванні є пам’ять, це і призвело до способу організації пам’яті з кількох сегментів, кожний з яких має своє функціональне призначення: сегмент кодів, сегмент даних, сегмент стека, кілька додаткових сегментів даних залежно від моделі МП. МП I8086 та I8088 можуть працювати тільки у так званому реальному режимі, адресуючи 1 Мбайт пам’яті. Мікропроцесорам моделей від I80286 до Pentium 4 є доступні два режими функціонування − реальний та захищений. Реальний режим дозволяє процесору, навіть якщо це Pentium 4, адресувати також тільки 1 Мбайт пам’яті. При включенні живлення комп’ютера та при його повторному пуску всі моделі мікропроцесорів фірми Intel починають працювати саме у реальному режимі.

Таким чином, МП 8086 адресує пам’ять 1 Мбайт, яка являє собою масив з 2^{20} 8-розрядних комірок. У пам’яті можуть зберігатись як байти, так і двобайтові слова. Слова розміщуються у двох сусідніх комірках пам’яті за принципом little endian − старший байт розміщується за старшою адресою, а молодший − за молодшою. Адресою слова вважається адреса його молодшого байта. Організація пам’яті, коли кожній адресі відповідає вміст однієї комірки пам’яті, називається **лінійною**. У реальному режимі адреса пам’яті є комбінацією сегментної адреси та зміщення − двох логічних адрес. Сегментна адреса, задана одним із сегментних регістрів, визначає початок сегмента пам’яті обсягом 64 кбайт. Зміщення задає положення комірки пам’яті усередині сегмента у байтах відносно до початку сегмента, тобто адресування усередині сегмента є лінійним. Складові лінійної адреси − сегмент та зміщення − часто записують парами через двокрапку: CS:IP, SS:SP.

Розмір сегмента становить 64 кбайти. Сегменти можуть бути суміжними, розділеними, перекриватися повністю або частково.

Початкова адреса, тобто найменша адреса у цьому сегменті, вміщується у відповідні сегментні регістри, може установлюватися робочою програмою та завжди починається з 16 байтової межі, наприклад, 0000H, 7000H.

Сегментні регістри мають 16 розрядів і доступні програмно через команди пересилань типу MOV та POP. Вони мають цільове призначення і не

можуть виконувати функції регістрів загального призначення в арифметичних і логічних операціях.

Сегментний регістр команд CS – керуючий регістр, що вказує сегмент, який вміщує адресу поточно виконуваної команди. Його вміст може бути змінений тільки при виконанні команд переходів, зверненні до підпрограм, поверненні з підпрограм або перериваннях із зазначенням іншого сегмента команд.

Сегментний регістр даних DS – керуючий регістр, що вказує сегмент, який вміщує програмно змінювані таблиці, масиви даних та константи. Така організація, за якої дані групуються разом в окремому сегменті і входять до сегмента, що зберігає коди команд, полегшує програмування. Усі прямі та непрямі засоби адресування даних, що використовують регістри BX, SI або DI, визначаються відносно регістра DS.

Сегментний регістр стека SS – керуючий регістр, вміст якого вказує початок стекової пам’яті (вершину). Вміст регістрів SP або BP може використовуватись для зазначення відносних адрес переходів, які формуються за участю цих регістрів, або адресу відносно регістра SS.

Сегментний регістр додаткового сегмента даних ES, а також сегментні регістри додаткових сегментів у МП I80386 та старших моделях FS та GS, – керуючі регістри. Вміст цих регістрів вказує на початок області пам’яті, яка зазвичай використовується для запам’ятовування проміжних результатів, тобто робочої області пам’яті. За необхідності використання додаткових сегментних регістрів використовуються команди з префіксом заміни сегмента, наприклад, команда

```
MOV AX,  ES:[100]
```

звертається до додаткового сегмента даних, початкова адреса якого знаходиться у регістрі ES.

Вказівник команд IP – регістр, що виконує роль лічильника команд. Його вміст вказує адресу наступного байта команди у сегменті пам’яті, що визначається вмістом регістра сегмента команд CS. Вміст регістрів IP та CS однозначно визначає адресу байта в усьому адресному просторі. Вміст регістра IP можна змінити тільки при виконанні команд переходів, виклику та поверненні з підпрограм та перериваннях.

Мікропроцесор I8086 є 16-розрядний, з 16-розрядними внутрішніми шинами адрес та даних, 16-розрядними регістрами та АЛП. З метою підвищення частоти функціонування МП, яка обмежується також кількістю виводів ВІС, зовнішня шина адреси AD0…AD15 даних є мультиплексована і також 16-розрядна. Окремо є 4 виводи ВІС A16...A19, які у багатопроцесорному максимальному режимі дозволяють виводити адресні розряди A16...A19 або інформацію про використовуваний у програмі на цей час сегментний регістр (ES, SS, CS, DS) та стан прапорця IF дозволу переривань.

Сегментні регістри мають довжину 16 розрядів, зміщення адреси даного у сегменті відносно початкової адреси сегмента (так звана ефективна або

виконавча адреса) має також 16 розрядів, а фізична адреса, тобто адреса комірки пам'яті в єдиному адресному просторі МП системи у реальному режимі, що видається на шину адреси, вміщує 20 розрядів, бо треба адресувати 1 Мбайт пам'яті. Фізична адреса усіх МП сім'ї INTEL у реальному режимі складається з суми вмісту одного з сегментних регістрів, зсунутого вліво на чотири розряди, та ефективної адреси або IP, SP, BP (рис. 9.1)

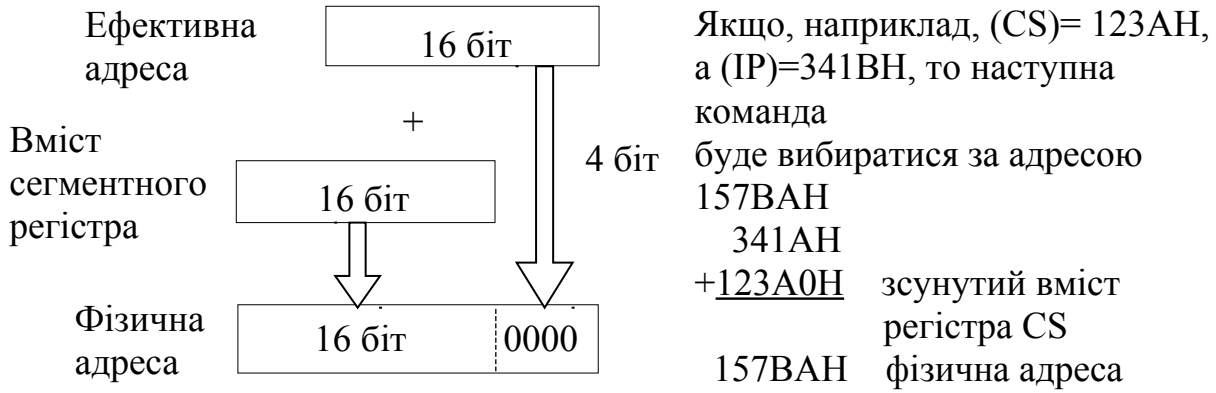


Рисунок 9.1 – Принцип формування 20-розрядної фізичної адреси

Формування фізичної адреси в усіх моделях до I80386 здійснюється за схемою, що подана на рис. 9.2.

Ефективна адреса є сумою вмісту вказаного у команді регістра BX, BP, DI, SI та поданого безпосередньо у формі числа зміщення, або сумою деякої бази, що зберігається у базовому регістрі BX або BP, вмісту індексного регістра DI або SI та зміщення.

Перенесення зі старшого біта, яке може виникнути при обчисленні фізичної адреси комірки пам'яті, ігнорується. Це призводить до кільцевої організації пам'яті: за коміркою з максимальною адресою FFFFF йде комірка з нульовою адресою. Аналогічну кільцеву структуру має і кожний сегмент. Таким чином, фізична адреса комірки пам'яті однозначно визначається 20-бітовим числом у діапазоні 0 − FFFFF у просторі пам'яті 1 Мбайт. Коди команд та даних можна вільно розміщувати за будь-якою адресою, що дозволяє економити пам'ять завдяки її ущільненню. Але для підвищення швидкості виконання програми доцільно розміщувати слова даних за парними адресами, тому що МП передає такі слова за один цикл шини. Для передачі слова з непарною адресою потрібні два цикли шини, що знижує продуктивність МП. Шинний інтерфейс ініціює необхідну для вибирання слова кількість звернень до пам'яті автоматично, тобто дворазове звернення до пам'яті не потребує спеціальних вказівок у програмі. Вказівник стека SP треба завжди ініціалізовувати на парну адресу, тому що обмін з пам'яттю відбувається тільки словами. Команди завжди вибираються словами за парними адресами. Виняток становить перша вибірка після передачі керування за непарною адресою, коли вибирається один байт. Вирівнювання команд за парною адресою не впливає на продуктивність МП, тому що при заповненні конвеєра (черги команд) вони поділяються на байти.

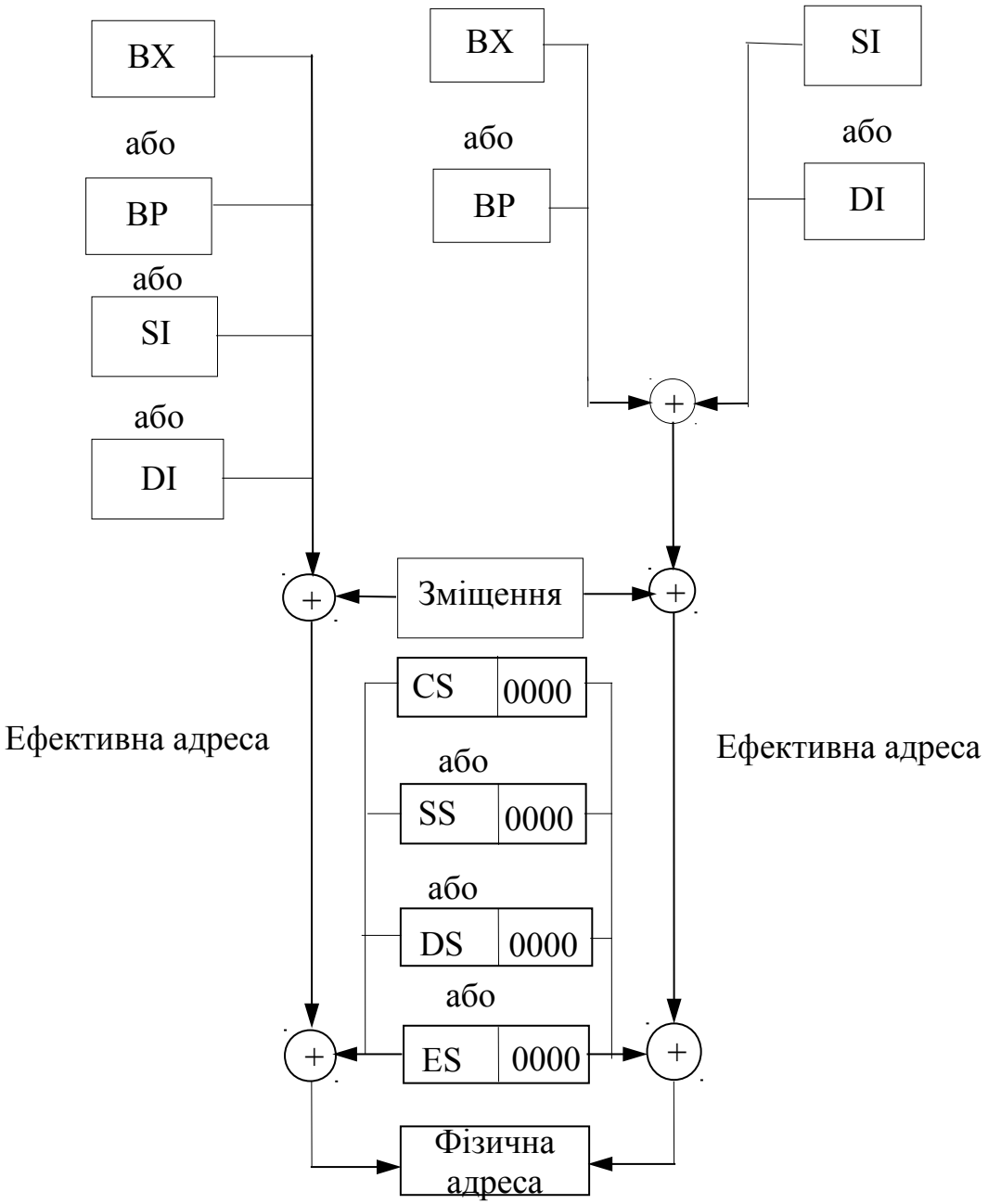


Рисунок 9.2 – Способи формування фізичної адреси

Сегментна організація пам’яті характеризується тим, що програмно доступною є не вся пам’ять, а лише організовані сегменти.

У сучасних 32-розрядних МП розмір сегмента може сягати 4 Гбайт.

У сегментах команд розміщено коди команд і при виконанні програми з цього сегмента інформацію можна тільки зчитувати, що є одним зі способів захисту кодів команд. У решті сегментів розміщуються дані, які у процесі виконання програми можна як зчитувати, так і записувати, тобто сегменти мають різні права доступу, а також різне змістовне призначення. При адресуванні операндів у командах з різними способами адресації не треба вказувати кожного разу вміст відповідного сегментного регістра, а оперувати тільки ефективною адресою, що скорочує команду та економить пам’ять.

Механізм сегментування пам’яті, незважаючи на складність, спрощує розподіл пам’яті при завантаженні програм. Програми, написані для роботи у реальному режимі, можуть працювати також у захищеному режимі старших моделей МП фірми Intel. Код та дані при роботі ПК або МПС з багатозадачною операційною системою можуть бути розміщені у довільних областях пам’яті без змін у програмі, що дозволяє використовувати програмне забезпечення на ПК з різною конфігурацією пам’яті.

Оскільки у межах сегмента дані адресуються зміщенням, сегмент можна пересувати у будь-яку область адресного простору без корекції зміщень: команда копіюється як єдиний блок пам’яті у нову область, після чого коректуються значення сегментних регістрів.

Кожна програма може складатись з будь-якої кількості сегментів, але безпосередній доступ без перевантаження сегментних регістрів вона має тільки до сегментів коду, даних, стека та одного додаткового сегмента даних для I8086. Програма ніколи не знає, за якими фізичними адресами будуть розміщені її сегменти, це вирішує операційна система.

Механізм керування пам’яттю є апаратний, тобто у реальному режимі програма не може сама сформувати фізичну адресу пам’яті, яка видається на шину адреси. Результат використання цього механізму є такий:

– компактність зберігання адреси у машинній команді;
– гнучкість способів адресування;
– захист адресних просторів задач у багатозадачній системі;
– підтримка віртуальної пам’яті при роботі у захищеному режимі.

Старші моделі МП фірми Intel, починаючи з I80386, підтримують апаратно кілька моделей використання оперативної пам’яті:

– сегментовану модель;
– сторінкову модель, яку можна розглядати як розташовану поверх сегментованої моделі; у рамках цієї моделі оперативна пам’ять поділяється на блоки фіксованого розміру 4 кбайт; сторінкова модель використовується при організації віртуальної (надаваної) пам’яті, що дозволяє операційній системі використовувати для роботи програм простір пам’яті більший, ніж обсяг фізичної оперативної пам’яті (до 4 Тбайт для МП Pentium).

Недоліками сегментної організації пам’яті є:

– сегменти безконтрольно розміщуються з будь-якої адреси, кратної 16 (вміст сегментного регістра апаратно зміщується ліворуч на 4 двійкових розряди);
– максимальний розмір сегмента для МП I8086 становить 64 кбайт;
– сегменти можуть перекриватись з дозволу програміста.

Ці недоліки зумовили застосування захищеного режиму у старших моделях МП.

9.2 Способи адресування операндів МП фірми Intel

Регістрове адресування операндів

Найбільш швидко виконуваним є регістрове адресування, за якого обидва операнди знаходяться у регістрах, які вказуються у команді. Вміст регістра-джерела передається (копіюється) у регістр-приймач, який у команді вказується першим. Наприклад:

```
MOV AX, CX; Копіювання 16-розрядного вмісту регістра CX в
          ; акумулятор AX
```

Вміст регістра-приймача завжди змінюється на вміст регістра-джерела, який не змінюється.

Безпосереднє адресування операндів

При безпосередньому адресуванні операнд-джерело вказується безпосередньо у команді у вигляді 8- або 16-розрядного даного. Операнд-приймач може бути регістром, або коміркою пам'яті, адресованою будь-яким способом, але не може бути числом. Наприклад:

```
MOV BL, 11H; 8-розрядне дане записується у регістр BL
```

При безпосередньому адресуванні знак константи поширюється до 8- або 16-розрядного регістра, а від'ємні числа записуються у регістр або комірку пам'яті у доповняльному коді.

Припустимо, що до виконання команди

```
MOV CL, 12H; Завантаження у CL константи 12H
```

у регістрі CL було записане число FFH (11111111B). Після виконання цієї команди у регістр CL буде записано число 12H (00010010B).

Результатом виконання команди

```
MOV CH, -30; Завантаження у CH константи -30D
```

буде запис у регістр CH числа E2H (11100010B) – числа -30D у доповняльному коді.

Покажемо на прикладі команди з безпосереднім адресуванням як розміщуються байти команди у сегменті кодів. При уведенні команди

```
MOV AX, 7000H
```

наприклад, у сегмент кодів, який починається з адреси 7000H (CS) та зміщення 0100H три байти команди у машинних кодах будуть розміщені у трьох послідовних комірках пам’яті з адресами

Адреса	Машинні коди	
7000:0100	B8	; Код операції “пересилати до AX”
7000:0101	00	; Молодший байт
7000:0102	70	; Старший байт операнда безпосередньо
7000:0103		; Вільна комірка пам’яті для прийому коду
		; операції наступної команди

Запис команди у сегмент кодів можна також проілюструвати за допомогою фрагмента пам’яті (рис. 9.3):

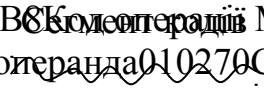


Рисунок 9.3 – Розташування байтів команди MOV AX, 7000H у сегменті кодів

Таким чином, обсяг пам’яті, яку займає програма, є сумою байтів, займаних усіма командами цієї програми.

Пряме адресування

При прямому адресуванні ефективна адреса операнда у сегментах даних вказується прямо у команді і розміщується у квадратних дужках. Початкова адреса сегмента даних, яка заздалегідь має бути розміщена у відповідному сегментному регістрі даних DS або ES, участь у формуванні ефективної адреси не бере. На рис. 9.4 показано фрагмент сегмента даних, в якому у комірках пам’яті з ефективними адресами 0010H та 0011H зберігаються відповідно дані BBH та AAH, а з адресами 0020H та 0021H – дані CCH та DDH.

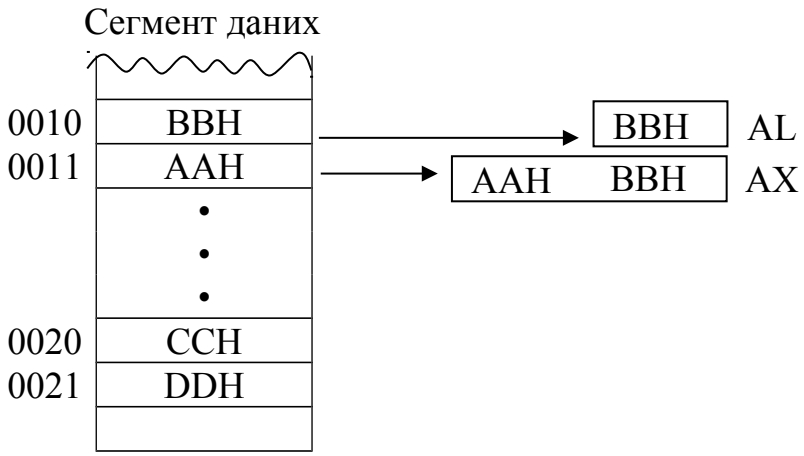


Рисунок 9.4 – Пряме адресування

Після виконання команди

```
MOV AL, [0010H]; Завантаження в AL даного з комірки пам'яті з адресою
                ; 0010H
```

у регістр AL буде завантажене дане BBH.

Після виконання команди

```
MOV AX, [0010H]; Завантаження в AX слова з комірок пам'яті з адресами
                ; 0010H та 00011H
```

у регістр AX буде завантажено слово AABBH відповідно до принципу зберігання у пам'яті слів – little endian.

Після виконання фрагмента програми

```
MOV AX, [0010H];
MOV [0020H], AX; Завантаження комірок пам'яті з адресами 0020H та
                ; 0021H з акумулятора AX
```

у комірках пам'яті з адресами 0020H та 0021H будуть записані нові дані: BBH та AAH відповідно.

Непряме регістрове адресування

При непрямому регістровому адресуванні ефективна адреса вміщується у будь-якому базовому або індексному регістрі: BX, BP, SI, DI. Непряме регістрове адресування стає у нагоді при обробленні елементів масивів. На рис. 9.5 показані два масиви, які розташовані у сегменті даних, починаючи

відповідно з адрес 0010H – перший та 0020H – другий, елементи яких становлять відповідно числа 12H, 34H, 56H, 78H та A1H, A2H, A3H, A4H.

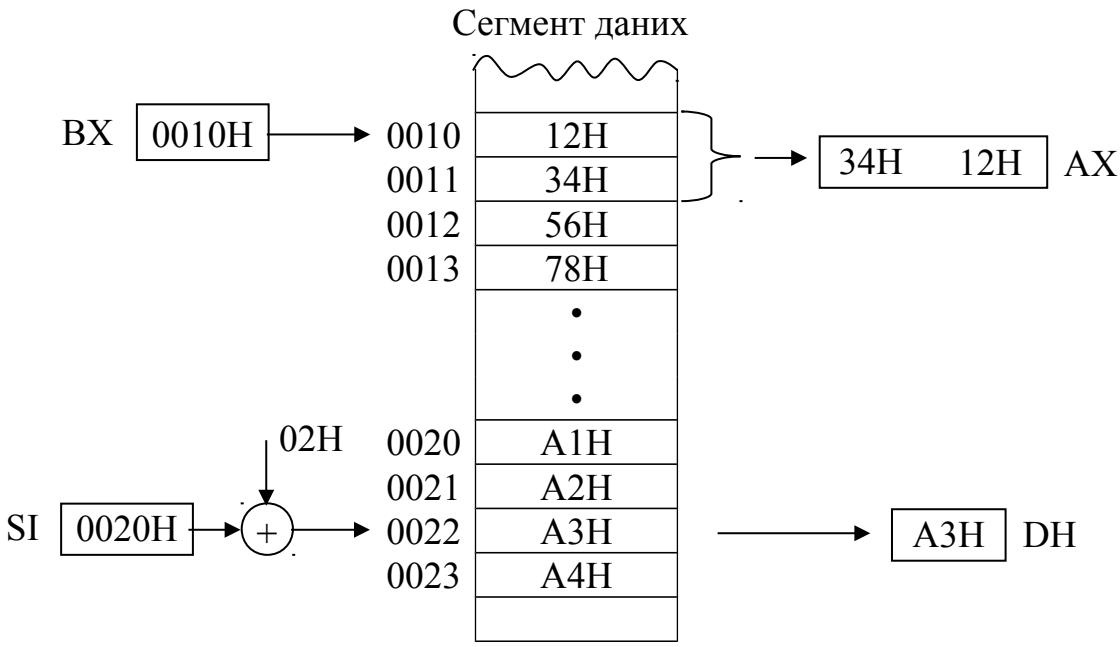


Рисунок 9.5 – Непряме регістрове адресування

Після виконання фрагмента програми

```
MOV BX, 0010H
MOV AX, [BX]
```

у регістр AX з двох комірок пам’яті з адресами (BX) та (BX+1) будуть передані два байти: 12H у регістр AL та 34H у регістр AH.

Після виконання фрагмента програми

```
MOV DI, 0022H
MOV DH, [DI]
```

у регістр DH буде записане дане A3H.

Спосіб непрямого регістрового адресування використовується для послідовного доступу до елементів масивів при нарощуванні або зменшенні вмісту базових або індексних регістрів. Якщо елементи масиву трактуються програмістом як двобайтові (перший фрагмент), то вміст базового або індексного регістра треба змінювати на 2, а якщо як однобайтові (другий фрагмент), то на 1.

При непрямому регістровому адресуванні при зазначенні адреси комірки пам’яті до вмісту базового або індексного регістра може додаватись зміщення.

Ефективна адреса підраховується як алгебраїчна сума вмісту вказаного у команді регістра та зміщення. Результатом виконання фрагмента програми

```
MOV SI, 0020H
MOV DH, [SI+0002H]
```

буде запис у регістр DH однобайтового елемента другого масиву з комірки пам’яті з ефективною адресою

$$EA = 0020H + 0002H = 0022H,$$

тобто даного A3H.

Такий спосіб адресування називається ще **непрямим регістровим адресуванням зі зміщенням**.

Адресування за базою

За цим способом ефективна адреса операнда обчислюється шляхом підсумовування вмісту базових регістрів BX або BP та зміщення – 8- або 16-розрядного числа. Регістри бази використовуються для доступу до структурованих записів даних (масивів), розташованих у різних областях пам'яті. Базова адреса масиву, тобто його початкова адреса у сегменті, розміщується у базовому регістрі, а доступ до її окремих елементів здійснюється за їх зміщенням відносно базової адреси. Для доступу до елементів різних масивів, розташованих на однаковій відстані від бази, достатньо змінити вміст базового регістра. Припустимо, що у сегменті даних, починаючи з ефективних адрес 0010 та 0020 розташовані відповідно два одновимірних масиви з елементами (рис. 9.6): 12H, 34H, 56H, 67H, 89H та A1H, A2H, A3H, A4H, A5H, A6H. За необхідності переслати у регістр AL четвертий елемент першого масиву, а у регістр AH – четвертий елемент другого масиву треба виконати такий фрагмент програми:

```
MOV BX, 0010H      ; Запис у BX базової адреси першого масиву
MOV AL, [BX+04H]; Пересилання до AL четвертого елемента першого
                   ; масиву (89H)
MOV BX, 0020H      ; Запис у BX базової адреси другого масиву
MOV AL, [BX+04H]; Пересилання до AH четвертого елементу другого
                   ; масиву (A5H)
```

Ефективна адреса четвертого елемента першого масиву становить 14H, а четвертого елемента другого масиву – 24H. Слід зазначити, що наступні команди:

```
MOV AX, [BX]+4
MOV AX, 4[BX]
MOV AX, [BX+4]
```

з точки зору підрахування ефективної адреси операнда у сегменті даних є ідентичними і відрізняються тільки формою завдання адреси.

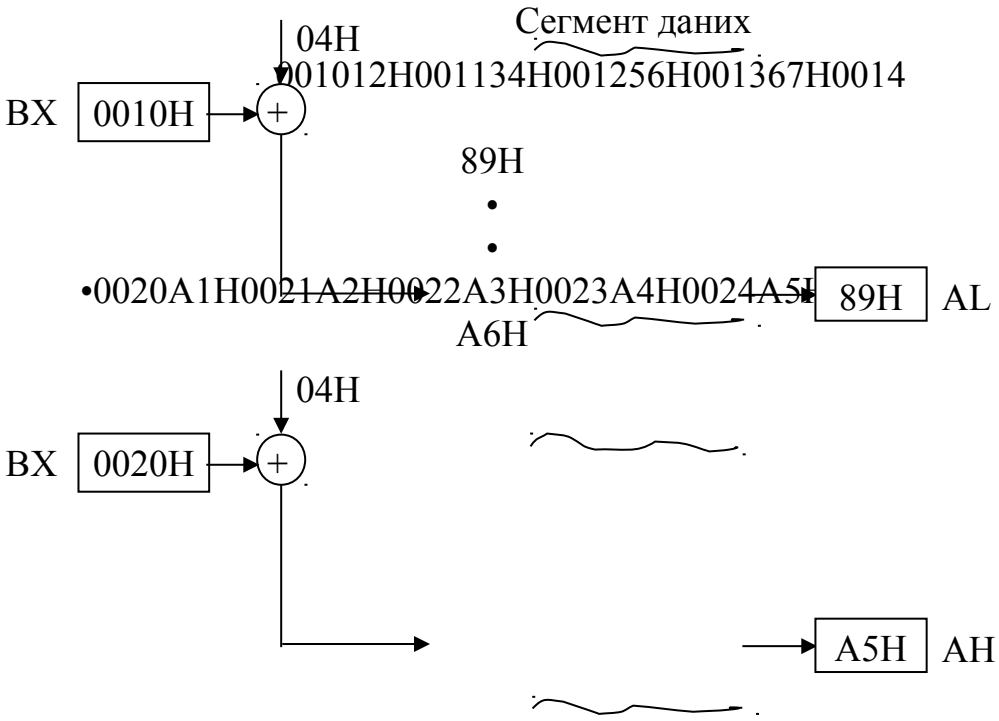


Рисунок 9.6 – Адресування за базою

Коли елементи масиву трактуються як двобайтові, як у наведених командах, то у регістр AX буде завантажено при значенні вмісту BX, який дорівнює 0020H, другий елемент другого масиву, тобто A6A5H.

Пряме адресування з індексуванням

При прямому адресуванні з індексуванням ефективна адреса обчислюється як сума зміщення (прямої адреси) та вмісту індексного регістра. Цей тип адресування зручний для доступу до елементів масиву у сегменті даних, коли прямо адресується початок масиву, а вміст індексного регістра вказує на потрібний елемент (рис. 9.7). Фрагмент програми:

```
MOV SI, 0002H
MOV AX, [SI+20H]
```

призведе до пересилання з сегмента даних у регістр AX двобайтового даного ACBAH з комірок пам'яті з адресами [20H+2H] та [20H+2H+1H].

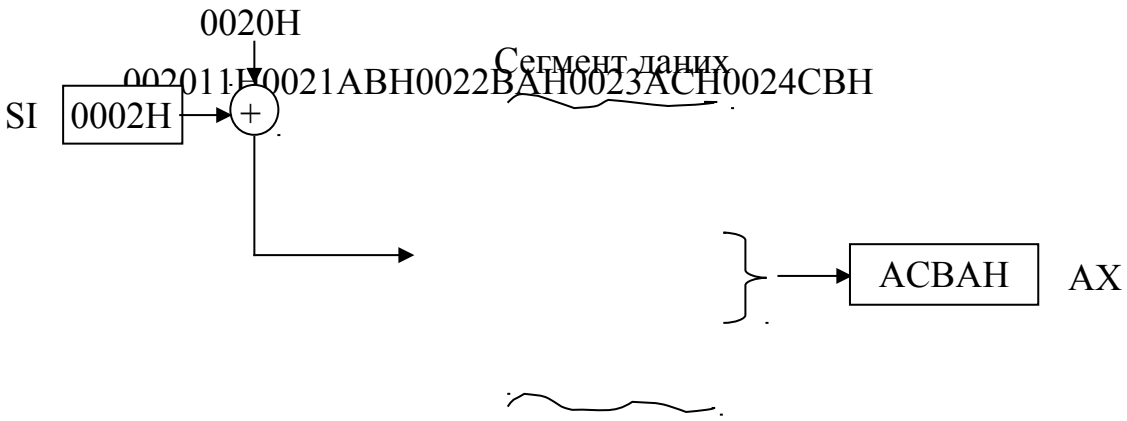


Рисунок 9.7 – Пряме адресування з індексуванням

Адресування за базою з індексуванням

При адресуванні за базою з індексуванням ефективна адреса обчислюється як сума значень базового регістра, індексного регістра і, можливо, зміщення. Такий спосіб адресування зручно використовувати для адресації двовимірних масивів, коли базовий регістр вміщує початкову адресу масиву, а значення зміщення та вміст індексного регістра вибирають елементи у рядку та стовпчику. Наступні команди є ідентичні:

```
MOV AX,[BX+2+DI]  ; Пересилання у регістр AX
                  ; елемента, адреса якого
MOV AX,[DI+BX+2]  ; визначається вмістом регістра DX
                  ; з третього
MOV AX,[BX+2][DI] ; стовпчика масиву, початкова адреса
                  ; якого
MOV AX,[BX][DI+2] ; знаходиться у регістрі BX
```

Команди з таким типом адресування виконуються найдовше.

Непряме адресування з масштабуванням

У старших моделях мікропроцесорів фірми Intel (I80386 та вищих) використовується також непряме адресування з масштабуванням. Зміщення задається двома 32-розрядними регістрами (базовим та індексним). Вміст індексного регістра множиться на масштабний множник 1, 2, 4 або 8. Масштабний множник використовується для адресування елементів масивів слів, множник 4 – для обробки подвійних слів, множник 8 – для доступу до елемента масиву 8-байтових даних. Наприклад, команда

```
MOV EAX, [EBX+4*ECX]
```

копіює в EAX подвійне слово з сегмента даних, розташоване, починаючи з адреси 4*ECX+EBX.

Контрольні запитання:

1 З якою метою організовується сегментування основної пам'яті МП сім'ї Intel?

2 Який обсяг пам'яті можуть адресувати МП сім'ї Intel у реальному режимі?

3 Як адресується сегмент основної пам'яті МП сім'ї Intel?

4 Яке призначення за умовчанням має кожний із сегментних регістрів мікропроцесорів фірми Intel?

5 Як обчислюється фізична адреса комірок основної пам'яті МП сім'ї Intel?

6 Який спосіб адресування у командах реалізується найшвидше?

7 Які особливості має безпосереднє адресування операндів?

8 В яких випадках операнд у командах адресується прямо?

9 Де вміщується ефективна адреса операнда при непрямому регістровому адресуванні?

10 Які способи адресування використовуються для звернення до елементів масиву?

11 Які особливості має непряме адресування з масштабуванням?

12 Які переваги має сегментування пам'яті порівняно з лінійною організацією?

13 Які недоліки має сегментування пам'яті?

14 Виконати фрагмент програми:

1) Задати у пам'яті, починаючи з ефективних адрес 0010H та 0020H, два масиви по 10H елементів кожний.
2) Завантажити сегментний регістр даних початковою адресою 7000H.
3) Завантажити індексний регістр SI константою 04H.
4) Використовуючи адресування з індексуванням, завантажити регістри DH та CX третім елементом першого масиву. Вказати вміст регістрів після завантаження.
5) Використовуючи адресування з індексуванням, завантажити регістри CL та DX другим елементом другого масиву. Вказати вміст регістрів після завантаження.

Контрольні запитання підвищеної складності:

1 У пам'яті розташований двовимірний масив 3 x 3 елементи у вигляді квадратної матриці; використовуючи непряме регістрове адресування зі зміщенням та індексуванням, вкажіть адресу центрального елемента масиву?

2 Покажіть, як будуть змінюватись адреси елементів цього масиву під час руху вдовж головної діагоналі матриці.

9.3 Мова програмування Асемблер-86

Вхідний контроль:

1 Що входить до поняття “архітектура мікропроцесорів”?

2 В чому полягає різниця між мовами програмування високого і низького рівнів?

3 Як можуть будуть подані дробові числа в обчислювальній системі?

4 В яких кодах подаються числа зі знаком в обчислювальній системі?

5 Наведіть регістрову модель 16-розрядного МП фірми Intel.

6 Які сегменти існують в основній пам’яті МПС, побудованої на МП фірми Intel?

7 Які способи адресування операндів МП фірми Intel Ви знаєте?

8 Що називається стеком і як його організовано?

Кожна мікропроцесорна система має свою власну мову програмування – мову машинних команд (машинну мову) і функціонувати може безпосередньо лише під керуванням програми, яка написана цією мовою. Машинна мова є сукупністю двійкових кодів усіх операцій, які може виконувати певний процесор. Машинна мова є незручною для широкого використання, тому що потребує досить великих витрат часу для написання і налагодження таких програм. Більш широке поширення находять мови програмування, які не співпадають з машинною мовою і є зручнішими за неї при використанні. Усі мови програмування можливо поділити на дві групи: мови високого і низького рівнів (машинно-орієнтовані мови), що відбиває ступінь близькості цих мов до машинної.

Мова Асемблер – це машинно-орієнтована мова, яка дозволяє використовувати усі структурні особливості мікропроцесорної системи, що пов’язані з апаратними можливостями, набором машинних команд, складом периферійного обладнання тощо. Мова Асемблер – це символічне подання машинної команди у вигляді її мнемонічного зображення. Програмування на Асемблері вимагає знання способів подання й оброблення даних на рівні машинних команд, що забезпечується знанням різних систем числення та архітектури МПС. Мова Асемблер поєднує у собі переваги машинної мови і деякі особливості мов високого рівня. Асемблер є найбільш ефективною мовою програмування при розв’язанні задач розробки системного програмного забезпечення, розробки програм для обміну даними з нестандартним периферійним обладнанням (драйвери), програмуванням систем реального часу для керування технологічними процесами й обладнанням, а також для забезпечення роботи інформаційно-обчислювальних систем та ефективного використання можливостей (робота у захищеному режимі) і ресурсів МПС.

Мова Асемблер мікропроцесорів фірми Intel є досить розвиненою і гнучкою. До складу мови Асемблер-86 входять понад 100 базових команд, відповідно до яких генерується понад 3800 машинних команд; близько 20 директив, призначених для розподілення пам’яті, ініціалізації змінних, умовного асемблювання тощо. Також у ній передбачено використання засобів

структурування даних, що є характерним для мов програмування високого рівня.

Всі команди мови Асемблер можливо поділити за групами, відповідно їх функціональному призначенню: пересилання даних, арифметичні операції, логічні операції і зсуви, передачі керування, оброблення рядків даних і команди керування станом центрального процесора.

Сукупність команд і директив, що представляють текст програми, називаються **початковим модулем**. Початкові модулі створюються за допомогою текстового редактора і можуть зберігатися на будь-якому носії у вигляді початкового файла. Цей файл може мати будь-яку назву і розширення .asm. Асемблер перетворює початковий модуль в **об'єктний модуль**. Ця операція називається **трансляцією**, а програма, яка її виконує – **транслятором**. На етапі створення об'єктного модуля виконується перетворення команд Асемблера на машинні команди. У результаті роботи транслятора створюються: об'єктний модуль – файл, який має розширення .obj, файл лістинга, який має розширення .lst і файл перехресних посилань з розширенням .crf. Файл лістинга вміщує код Асемблера початкової програми, для кожної команди якої вказано машинний код і зміщення у кодовому сегменті, крім того, до цього файла входять таблиці з інформацією про мітки і сегменти, які використовуються у програмі і повідомлення про синтаксичні помилки. Імена файлів призначає програміст, вони можуть співпадати, а можуть бути різними.

Після створення об'єктного модуля він оброблюється за допомогою програми-укладача, яка генерує завантажувальний модуль. Це файл, який має розширення .exe і який може виконуватися МПС. Програма-укладач в завантажувальному модулі об'єднує об'єктні модулі, що входять до програми і підключає, за необхідності, бібліотечні модулі і замінює відносні адреси комірок пам'яті на абсолютні, з урахуванням області пам'яті, в яку модуль буде завантажено для виконання. Бібліотечні модулі – це об'єктні файли, які містять найбільш поширені програми. Ім'я завантажувального модуля також призначає програміст. Блок-схему алгоритму підготовки програми на мові Асемблер для її виконання показано на рис. 9.8.

Текст програми на мові Асемблер складається з операторів (речень) чотирьох типів:

- команди, які є символічними аналогами машинних команд МПС;
- макрокоманди – конструкції, які при трансляції програми замінюються на послідовність відповідних команд;
- директиви – інструкції транслятору з виконання певних дій;
- коментарі – пояснення з виконання команди або фрагмента програми, використовуються для документування і кращого розуміння змісту програми; коментар ігнорується асемблером, тому він може бути написаний будь-якою мовою, і відокремлюється символом ; від іншої частини речення.

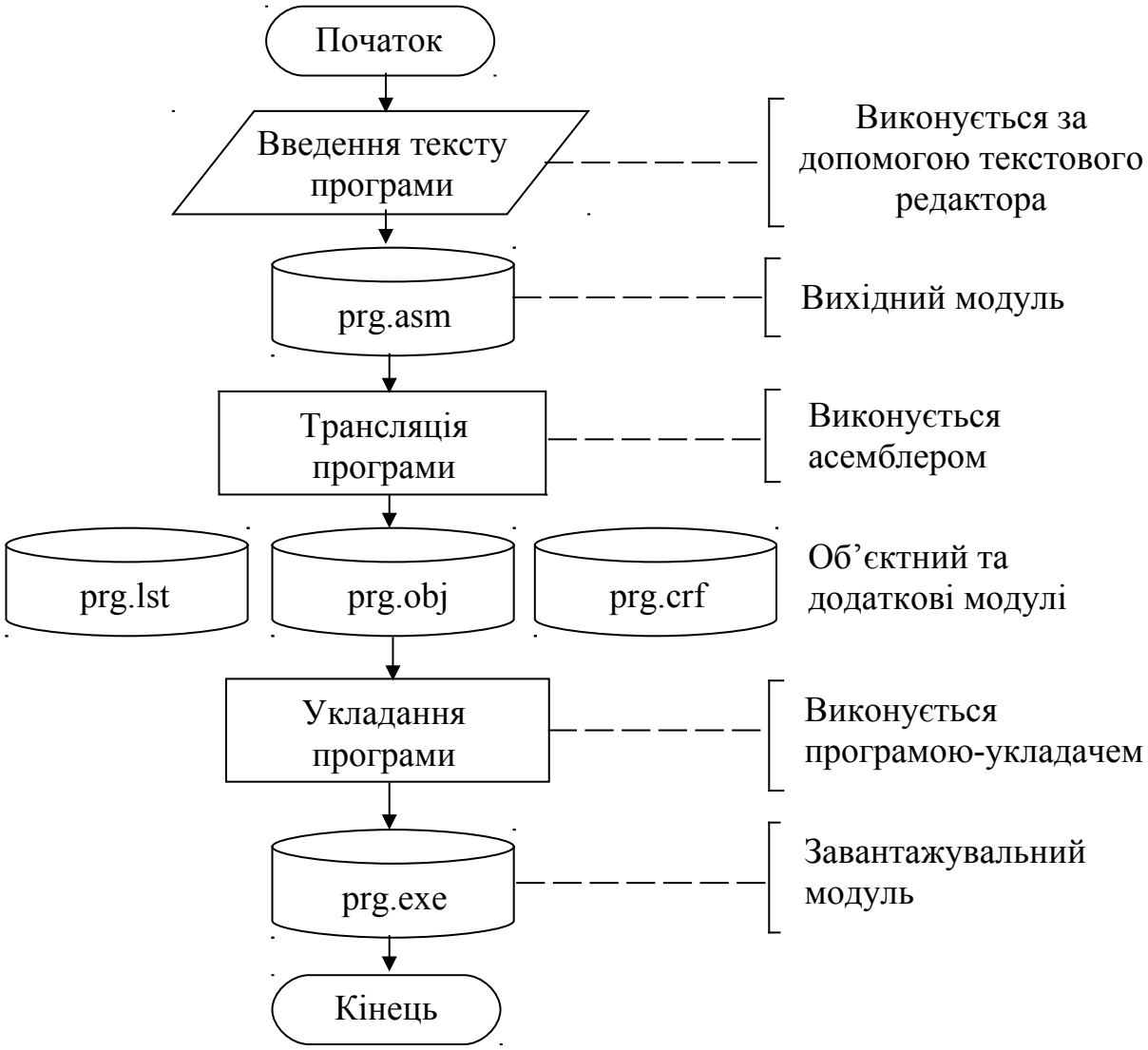


Рисунок 9.8 – Блок-схема підготовки програми для її виконання

Довжина речення становить 131 символ, усі інші ігноруються. Речення повинно розташовуватися у межах одного рядка, переносити речення на інший рядок не дозволяється. Речення Асемблера формуються із найпростіших конструкцій мови – **лексем**, синтаксично нероздільних послідовностей допустимих символів алфавіту Асемблера, які мають значення для транслятора. Як символи алфавіту Асемблера використовуються:

– усі літери латинського алфавіту, при цьому великі і малі літери є еквівалентні;

– цифри від 0 до 9;

– спеціальні і розподілювальні символи, до яких відносяться: ?, @, $, _, &, [ ], ( ), < >, { }, +, –, /, *, %, !, “ “, \, =, #,^ і деякі недруковані символи.

До лексем відносяться: ідентифікатори, рядки символів, цілі числа двійкової, шістнадцяткової або десяткової систем числення.

Ідентифікатор – це послідовність символів, яка визначена програмістом і використовується для іменування об’єктів програми (міток, змінних, назв операцій тощо). Ідентифікатор може складатися з одного або кількох символів, але починатися може лише літерою. Крапка може бути лише першим символом

ідентифікатора. Інші спеціальні символи використовувати в ідентифікаторах не дозволяється. Довжина ідентифікатора становить до 255 символів, але транслятор сприймає лише 32 перших символи, інші ігноруються.

Ідентифікатори поділяються на **службові слова** й **імена**. До службових слів відносяться ідентифікатори, значення яких однозначно визначено (назви регістрів, команд і таке інше). Імена – це символічні назви, які користувач призначає певним об'єктам програми (змінним, коміркам пам'яті тощо).

Рядком символів може бути будь-яка послідовність символів, в якості яких можливо використовувати літери і спеціальні символи, крім символа повернення каретки та перенесення рядка. Рядок символів обов'язково брати у лапки.

Через те, що мікропроцесор може працювати з різними сегментами, то для завантаження і роботи програми необхідно задати сегменти й особливості їх використання. Це робиться за допомогою директиви SEGMENT, яка має вигляд:

{Ім'я сегмента} SEGMENT {Тип вирівнювання} {Тип комбінування} {Клас сегмента} {Тип розміру}

Програма {Ім'я сегмента} END

Розглянемо окремі поля цієї директиви.

Тип вирівнювання – повідомлення програмі-укладачу про адресу початку сегмента. При правильному розміщенні програма виконується швидше. Можливі значення:

- BYTE – вирівнювання не виконується. Початкова адреса сегмента може бути будь-якою;
- WORD – сегмент починається з адреси, яка є кратна 2, при цьому молодший значний біт фізичної адреси дорівнює 0. Виконується вирівнювання по межі слова;
- DWORD – сегмент починається з адреси, яка є кратна 4. Виконується вирівнювання по межі подвійного слова;
- PARA – сегмент починається з адреси, яка є кратна 16, при цьому остання шістнадцяткова цифра фізичної адреси дорівнює 0H. Виконується вирівнювання по межі параграфа;
- PAGE – сегмент починається з адреси, яка є кратна 256. Виконується вирівнювання по межі сторінки;
- MEMPAGE – сегмент починається з адреси, яка є кратна 4 кбайтам.

За умовчанням тип вирівнювання має значення PARA.

Тип комбінування – повідомлення програмі-укладачу про об'єднання при виконанні сегментів різних модулів програми, які мають однакові імена. Можливі значення:

- PRIVATE – сегмент не об'єднується з іншими сегментами;

– PUBLIC – об’єднуються усі сегменти з однаковими іменами. Сегмент, що отримаємо, буде цілий і безперервний, а всі адреси будуть формуватися від його початку;

– COMMON – усі сегменти з однаковими іменами розміщуються за однією адресою. Таким чином, усі сегменти будуть перекриватися і сумісно використовувати пам’ять;

– AT xxxx – розміщує сегмент за абсолютною адресою параграфа, адреса якого задається виразом xxxx;

– STACK – визначення сегмента стека. Усі сегменти, що мають однакові імена, об’єднуються таким чином, що їх адреси розраховуються відносно вмісту регістра SS.

Клас сегмента – повідомлення про відповідний порядок включення сегментів при складанні програми із сегментів різних модулів. Програма-укладач з’єднує усі сегменти, що мають однаковий клас. Рекомендується об’єднувати сегменти з однаковим функціональним призначенням (наприклад, сегменти коду програми). Клас сегмента – це рядок, який береться у лапки, наприклад, сегмент коду «CODE».

Тип розміру сегмента – задає розмір сегмента і порядок формування фізичної адреси в ньому, тому що дані можуть бути 16- або 32-розрядними. Може приймати такі значення:

1 USE16 – формування фізичної адреси у такому сегменті виконується тільки з 16-розрядним зміщенням. Розмір такого сегмента 64 кбайт.

2 USE32 – формування фізичної адреси у такому сегменті виконується тільки з 32-розрядним зміщенням. Розмір такого сегмента 4 Гбайт.

Для призначення сегментам конкретного функціонального призначення і закріплення їх за сегментними регістрами використовується директива ASSUME, яка має вид:

ASSUME Назва сегментного регістра; Ім’я сегмента

Для простих програм, що мають по одному сегменту для коду, даних і стека, у найбільш поширених трансляторах TASM і MASM для спрощення директив сегментування уведено директиву зазначення моделі пам’яті MODEL, яка частково керує розміщенням сегментів. Ця директива зв’язує сегменти, які при використанні спрощених директив сегментування мають наперед визначені імена з сегментними регістрами. Обов’язковим параметром цієї директиви є модель пам’яті. У табл. 9.1 наведено назви і значення параметрів директиви MODEL.

У цій таблиці поняття near і far характеризують віддаленість об’єкта від місця використання. Якщо об’єкт розташовано у сегменті разом з кодом і звернутися до нього можливо за допомогою двобайтового зміщення, то це відповідає типу near (близький перехід). При цьому команда модифікує лише вміст вказівника команд IP. В іншому випадку об’єкт розміщується в іншому сегменті (тип far) і для звернення до нього необхідно надати чотирибайтний

вказівник – сегмент: зміщення. При цьому буде модифіковано вміст двох регістрів.

Таблиця 9.1 – Назви і параметри моделей пам’яті

Модель	Тип коду	Тип даних	Призначення моделі
TINY	near	near	Код і дані поєднано в одну групу з ім’ям DGROUP. Використовується для створення програм з розширенням .com
SMALL	near	near	Код займає один сегмент, дані об’єднано у одну групу з ім’ям DGROUP. Ця модель, звичайно, використовується для створення програм мовою Асемблера
MEDIUM	far	near	Код займає декілька сегментів, по одному на кожен програмний модуль, що об’єднується. Усі посилання на передачу керування – типу far. Дані поєднано в одній групі. Усі посилання на них – типу near
COMPACT	near	far	Код знаходиться в одному сегменті. Дані можуть бути у різних сегментах. Посилання на них типу far
LARGE	far	far	Код займає декілька сегментів, по одному на кожен програмний модуль, що об’єднується. Усі посилання типу – far

Опис простих типів даних. Для правильного описування і оброблення даних їх необхідно описати, для чого використовуються спеціальні директиви резервування й ініціалізації даних, які є вказівками транслятору на виділення певного обсягу пам’яті для розміщення даних певної довжини. Ці директиви мають вид:

- db – резервування пам’яті для даних довжиною 1 байт;
- dw – резервування пам’яті для даних довжиною 2 байта (слово);
- dd – резервування пам’яті для даних довжиною 4 байта (подвійне слово);
- df – резервування пам’яті для даних довжиною 6 байт;
- dq – резервування пам’яті для даних довжиною 8 байт;
- dt – резервування пам’яті для даних довжиною 10 байт.

Нижче наведено приклад використання директив для керування програмою на мові Асемблер.

```
MODEL SMALL                    ; Тип моделі пам’яті, що буде
                               ; використовуватися
STACK 256                      ; Визначення сегмента стека
```

```
DATA SEGMENT                          ; Визначення сегмента даних
DATA ENDS
CODE SEGMENT                          ; Визначення сегмента коду
ASSUME CS:CODE, DS:DATA, ES:DATA      ; Закріплення сегментних
                                      ; регістрів
Програма

CODE ENDS                             ; Кінець коду програми
END                                   ; Закінчення
```

Контрольні запитання:

1 Які особливості мови Асемблер забезпечують її використання?

2 Які файли формуються у результаті роботи програми-транслятора?

3 Чому виконання асемблерної програми можливо лише після роботи програми-укладача?

4 Які файли використовує програма-укладач для формування завантажувального модуля?

5 Які оператори (речення) можуть входити до тексту програми на мові Асемблера?

6 Які моделі пам’яті Ви знаєте? Чим відрізняються поняття near і far?

7 Які директиви використовуються для опису простих типів даних?

Контрольні запитання підвищеної складності:

1 Для чого використовуються різні типи комбінування?

2 Яким чином програмувач задає типи сегментів і особливості їх використання?

3 Які директиви резервування й ініціалізації даних Ви знаєте?

9.3.1 Формат команди

Формат рядка команди складається з кількох полів, деякі з них не є обов’язковими. Формат рядка команди показано на рис. 9.9. У фігурних дужках показано необов’язкові елементи.

Поле мітки	Поле команди	Поле операндів	Поле коментарів
{Мітка:}	{Префікс}Команда	{Операнди}	{;Коментар}

Рисунок 9.9 – Формат команди Асемблера

Мітка – це символічне ім’я, що відповідає значенню поточного вмісту лічильника адреси в поточному сегменті коду. Вміст лічильника адреси відповідає значенню регістра-вказівника команд (IP) при виконанні програми, тому мітка – це адреса команди, до якої необхідно зробити умовний або безумовний перехід. Мітка відокремлюється від інших частин рядка двокрапкою. Максимальна довжина мітки – 31 символ. Кожна мітка у програмі повинна бути унікальною. Як мітки не допускається використовувати службові слова.

Префікс – це елемент команди, який уточнює або модифікує її дію у випадках:

- зміна сегмента, якщо нас не задовольняє значення сегмента за умовчанням;
- зміна розрядності адреси;
- зміна розрядності операнда;
- вказівка на повторення цієї команди.

Команда (мнемокод) – службове слово, яке відповідає типу машинної команди, що генерується. Як команди використовуються скорочені (повні) англійські слова або абревіатури англійських слів, що передають значення основної функції команди. В залежності від подання операндів, для виконання однієї команди може генеруватися декілька кодів.

Операнди – імена, які надаються об’єктам, над якими виконується операція (команда). У команді можуть бути один або два операнди, які відокремлюються один від одного комою. Є команди, в яких операндів немає. Поле операндів відокремлюється від поля команди пропуском. У команді можливі лише такі сполучення операндів:

- регістр – регістр;
- регістр – пам’ять;
- пам’ять – регістр;
- безпосередній операнд (число) – регістр;
- безпосередній операнд (число) – пам’ять.

Як операнди, що можуть оброблюватися транслятором використовуються:

- постійні або безпосередні операнди – числа, рядки, імена або вирази, які мають фіксоване значення. Імена, які використовуються як безпосередні операнди, задаються операторами **=:** або equ;
- адресні операнди – адреси розміщення операндів у пам’яті. Задаються за допомогою двох складових адреси – сегмента і зміщення;
- переміщувані операнди – символьні імена, які є певними адресами пам’яті. Такі адреси вказують на розташування у пам’яті певних інструкцій (операнд – мітка) або даних (операнд – ім’я області пам’яті у сегменті даних). Переміщувані операнди відрізняються від адресних тим, що вони не зв’язані з конкретною адресою фізичної пам’яті, а формуються після завантаження програми для її виконання;

– лічильник адреси – це специфічний вид операнда, який позначається символом **$.** Транслятор замість цього символа ставить поточне значення вказівника команд і модифікує його значення відповідно до зміщення;

– регістровий операнд – ім'я відповідного регістра, вміст якого використовується у команді;

– базові і індексні операнди – операнди, які використовуються при адресуванні за базою, індексного адресування або їх комбінацій.

Загальний формат команди показано на рис. 9.10.

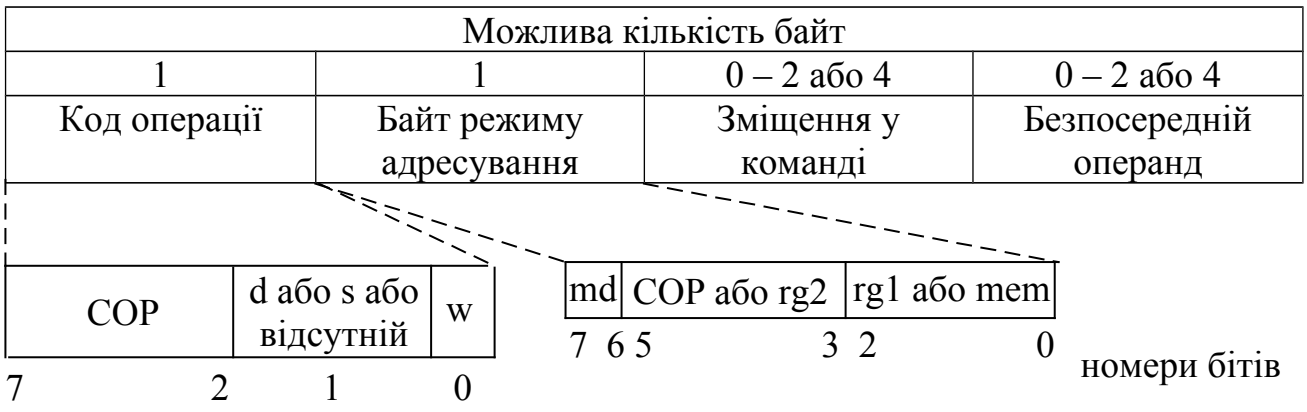


Рисунок 9.10 – Загальний формат команди Асемблера МП фірми Intel

Байт коду операції – це обов'язковий елемент, що описує операцію, яка буде виконуватися. Складається з поля коду (COP) операції і двох однобітових полів:

– w – відповідає типу операндів, які використовуються у команді:
  w = 0 – команда працює з байтами;
  w = 1 – команда працює зі словами;

– d – адресує приймач результату:
  d = 1 – відбувається передавання операнда або результату операції у регістр rg2, який описано у другому байті команди;
  d = 0 – відбувається передавання із цього регістра;

– s – біт, який є ознакою використання одного байта безпосереднього операнду при роботі зі словами. Використовується сумісно зі значенням w.

$$sw = \begin{cases} x0 – \text{один байт даних, який розміщується в молодшому байті операнду L;} \\ 01 – \text{два байта даних у двох байтах операнду H і L;} \\ 11 – \text{один байт даних, який поширюється зі знаком до слова.} \end{cases}$$

Байт режиму адресації визначає форму подання адреси операндів. Складається з трьох полів:

– md – визначає довжину адреси операнда (у байтах), яка подана у наступних байтах – зміщення у команді:

md = { 00 – зміщення у команді відсутнє й адреса операнда визначається вмістом регістрів BX, BP, DI, SI, які використовуються для обчислення фізичної адреси операнда;
01 – команда містить 8-розрядне зміщення у наступному байті, яке поширюється зі знаком до 16 розрядів;
10 – команда містить 16-розрядне зміщення у двох наступних байтах;
11 – операнди знаходяться лише у регістрах;

– COP або rg2 – значення COP використовується для уточнення операції, що виконується; rg2 визначає регістр, в якому знаходиться операнд, який умовно вважається другим;

– rg1 або mem – визначає операнд, який може бути розміщено у комірці пам'яті або у регістрі. Цей операнд умовно вважається першим. Кодування полів rg2 і rg1 в залежності від значення біта s подано у табл. 9.2.

Таблиця 9.2 – Кодування полів rg2 і rg1

rg2 / rg1	w = 0	w = 1	rg2 / rg1	w = 0	w = 1
000	AL	AX	100	AH	SP
001	CL	CX	101	CH	BP
010	DL	DX	110	DH	SI
011	BL	BX	111	BH	DI

Коментар – будь-який текст, який використовується для документування і кращого розуміння змісту програми. При трансляції програми коментарі ігноруються, тому наявність коментаря не впливає на ефективність виконання програми.

Контрольні запитання:

1 Який формат має рядок команди мови Асемблер?
2 Що називається операндом у рядку команди мови Асемблер?
3 Які поєднання операндів є можливими?
4 Який формат має команда Асемблера МП фірми Intel?
5 В яких байтах може розміщуватися поле коду операції?
6 За допомогою яких бітів кодується тип операндів, що використовуються у команді?

Контрольні запитання підвищеної складності:

1 Яким чином визначається тип режиму адресації у команді?
2 За допомогою яких полів визначаються регістри, які використовуються у команді?
3 Доведіть, що операнди команди MOV AX, BX розташовано у регістрах, якщо формат (код) цієї команди становить 89D8H.

9.3.2 Команди пересилань

Вхідний контроль:

1 Які програмно-доступні вузли МПС Ви знаєте?
2 Яким чином вказується адреса регістрів загального призначення?
3 Які регістри входять до складу центрального процесора?
4 Яким чином може вказуватися адреса комірок пам’яті?
5 Що таке сегментація пам’яті мікропроцесорів фірми Intel?

Усі команди, що входять до цієї групи, в свою чергу, зручно поділити на п’ять груп і розглядати кожну окремо. До групи команд пересилань входять:

– загальні команди пересилань;
– команди передавання даних з використанням стека (стекові команди);
– команди введення-виведення;
– команди передавання рядків даних;
– команди перетворення типів даних.

Усі команди пересилань не змінюють стан регістра прапорців, його вміст зберігається незалежно від виконання команди пересилань і кількості пересилань між будь-якими пристроями. Виключення становлять лише команди POPF і SAHF, які у подальшому будуть розглянуті.

Загальні команди пересилань – це команди, що здійснюють пересилання операндів: регістр – регістр, регістр – пам’ять, пам’ять – регістр, регістр – безпосередній операнд (число), пам’ять – безпосередній операнд (число).

Найбільш поширеною командою цієї групи є команда **MOV (MOVe operand)**, яка має вигляд

MOV *dst,src*.

Команда здійснює пересилання операнда *src* (джерело) до *dst* (приймач), причому вміст операнда *src* (джерело) не змінюється. У цій команді можливі усі способи адресації операндів, чим пояснюється її універсальність і розповсюдженість. При пересиланні даних між регістрами лише один з них може бути сегментним регістром. Типи даних, що пересилаються, повинні співпадати. В залежності від подання операндів команда може мати різний формат і займати від 3 до 6 байт.

Приклади цієї команди при використанні різних способів адресування:

```
MOV AL,BH      ; Пересилання байта з регістра BH до регістра AL
MOV BX,CX      ; Пересилання слова з регістра CX до регістра BX
MOV AL,[BX]    ; Пересилання байта з комірки пам’яті, адреса якої
               ; знаходиться у регістрі BX, до регістра AL
MOV AX,[BX]    ; Пересилання слова з двох сусідніх комірок
               ; пам’яті, молодша адреса котрих знаходиться у
```

; регістрі BX, до регістра AX

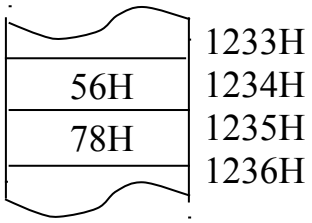
Виконання команди MOV AX,[BX] можливо пояснити таким чином: вважаємо, що до початку виконання цієї команди регістри і комірки пам'яті мали такий вміст:

Вміст регістра (BX) = 1234H – базова адреса комірки пам'яті у поточному сегменті даних.

Вміст регістра (DS) = 7000H – адреса поточного сегмента даних.

Вміст регістра (AX) – не визначений.

У комірках пам'яті поточного сегмента розміщені байти даних відповідно



Під час виконання операції відбувається копіювання вмісту комірки пам'яті з адресою, що знаходиться у регістрі BX, у молодший байт регістра AX – AL і комірки з наступною адресою (1235H) у старший байт регістра AX – AH. Після виконання цієї команди вміст регістрів буде:

Вміст регістра (BX) = 1234H – не зміниться.

Вміст регістра (DS) = 7000H – не зміниться.

Вміст регістра (AX) = 7856H – дані, які переслано з пам'яті.

Вміст комірок пам'яті не змінився.

Команда **MOVSX (MOVe and Sign eXtension)** – пересилання зі знаковим поширенням. Синтаксис команди:

MOVSX *dst, src*.

Використовується для перетворення операндів зі знаком, які мають малий розмір в еквівалентні елементи більшої розмірності.

Як операнди *dst* (приймач) використовуються 16- або 32-розрядні регістри, а операндом *src* (джерело) можуть бути 8- або 16-розрядні регістри або комірки пам'яті.

При виконанні команди вміст операнда "джерело" записується, починаючи з молодших розрядів в операнд "приймач", і знаковий розряд операнда "джерело" поширюється на вільні старші розряди операнда "приймач".

Наприклад, виконання команди

MOVSX AX,CL,

якщо до початку її виконання регістри мали такий вміст: (CL) = 82H, а (AX) – не визначено, полягає в копіюванні вмісту регістра CL у регістр AL і поширенні знакового розряду у регістрі AH. Після виконання команди у регістрі AX буде записано операнд FF82H.

Команда **MOVZX (MOVe and Zero eXtension)** – пересилання з нульовим поширенням. Синтаксис команди:

MOVZX *dst, src*.

Використовується для перетворення беззнакових операндів, які мають малий розмір в еквівалентні елементи більшої розмірності.

Як операнди *dst* (приймач) використовуються 16- або 32-розрядні регістри, а операндом *src* (джерело) можуть бути 8- або 16-розрядні регістри або комірки пам'яті.

При виконанні команди, вміст операнда "джерело" записується, починаючи з молодших розрядів в операнд "приймач", а у старші розряди операнда "приймач" поширюються нулі.

Наприклад, виконання команди

MOVZX AX,CL,

якщо до її виконання регістри мали такий вміст: (CL) = 82H, а (AX) – не визначено, полягає в копіюванні вмісту регістру CL у регістр AL і поширенню нулів у регістрі AH. Після виконання команди у регістрі AX буде записано операнд 0082H.

Команда **XCHG (eXCHanGe)** – обмін вмісту операндів. Синтаксис команди:

XCHG *dst, src*.

Використовується для обміну вмісту 8-, 16-, 32-розрядних регістрів між собою, або для обміну байтами, словами і довгими словами між регістрами і пам'яттю. Команда не працює з сегментними регістрами. Адресування пам'яті може виконуватися будь-яким способом.

Приклади цієї команди при використанні різних способів адресування:

```
XCHG AL,BH    ; Обмін вмісту (байта) регістра BH на вміст регістра AL
XCHG BX,CX    ; Обмін вмісту (слова) регістра CX на вміст регістра BX
XCHG AL,[BX]  ; Обмін байтом з комірки пам'яті, адреса якої
              ; знаходиться у регістрі BX, і вмістом регістра AL
XCHG AX,[SI]  ; Обмін словом з комірок пам'яті, адреса яких задана
              ; вмістом регістра SI і словом з регістра AX.
```

Команда **XLAT (transLATe byte from table)** – перетворення байта. Синтаксис команди:

XLAT.

Використовується для завантаження у регістр AL байта з 256-байтової таблиці, яка розміщена у пам'яті і початкова адреса якої знаходиться у регістрі BX. Вміст регістра AL у цій команді використовується як індекс у таблиці.

Команди **LDS/LES/LFS/LGS/LSS (Load pointer into DS/ES/FS/GS/SS segment register)** завантаження одного із сегментних регістрів вказівником з пам'яті. Синтаксис команд:

LDS/LES/LFS/LGS/LSS *dst, src*.

Як операнди *dst* (приймач) можуть використовуватись 16- або 32-розрядні регістри загального призначення або індексні, а як операнди *scr* (джерело) – 4 або 8 комірок пам'яті, адресація яких здійснюється довільним способом. Усі команди виконуються однаково, зміни обумовлені регістром, назва якого входить у команду.

Використовується для завантаження повного вказівника у вигляді сегментної складової та зміщення. Повний вказівник завантажується у регістр – приймач і сегментний регістр, який указаний в команді.

Виконання однієї з команд, наприклад

LES DI,[SI+2].

за умови попереднього стану:
(DS) = 7000H; (SI) = 1234; DX – не визначено; у комірках пам'яті з адресами 7000:1234 66 77 88 99 AA BB CC …
відбувається таким чином:

– вміст комірок пам'яті з адресами 1234+2 і (1234+2)+1 завантажуються у регістр DX. Його вміст стане – 9988H;

– вміст комірок пам'яті з адресами (1234+2)+3 і (1234+2)+2 завантажуються у регістр ES. Його вміст стане – BBAAH.

Команда **LEA (Load Effective Address)** – завантаження ефективної адреси. Синтаксис команди:

LEA *dst, src*.

Як операнди *dst* (приймач) можуть використовуватись 16- або 32-розрядні регістри загального призначення або індексні, а як операнд *src* (джерело) – ефективна адреса, яка обчислюється відповідно заданому режиму адресування.

В операнд завантажується значення операнда *src* (джерело), а не вміст комірки пам'яті, яку він адресує.

Команда використовується для завантаження ефективної адреси у регістр, вміст якого можливо вважати базовим при подальшій роботі, наприклад, при роботі з рядками даних.

Команди LDS/LES/LFS/LGS/LSS і LEA використовуються для ініціалізації регістрів МП перед виконанням обробки рядків даних за допомогою команд обробки рядків. Використання цих команд спрощує комутацію сегментів даних, тому що одночасно завантажується базовий або індексний регістр та сегментний регістр.

Приклад команди завантаження ефективної адреси:

```
LEA DX,[BX+SI+2]    ; Завантаження у регістр DX ефективної адреси,
                    ; значення якої становить суму вмісту регістрів
                    ; BX і SI та числа 2. Якщо вміст регістрів
                    ; (BX) = 0100H, (SI) = 0020H, то у регістрі DX
                    ; після виконання команди буде завантажено
                    ; число 0122H = 0100H+0020H+0002H
```

Команда **LAHF (Load AH register from register Flags) –** завантаження регістра AH ознаками результату із регістра FLAGS. Синтаксис команди:

LAHF.

Команда завантажує копію вмісту молодшого байта регістра Flags, в якому зберігаються 5 ознак – CF, PF, AF, ZF, SF, до 7, 6, 4, 2 і 0 бітів регістра AH.

Команда **SAHF (Store AH register into register Flags) –** завантаження регістра FLAGS із регістра AH. Синтаксис команди:

SAHF.

Команда відновлює вміст молодшого байта регістра Flags з регістра AH, в якому зберігалися 5 ознак результату – CF, PF, AF, ZF, SF у бітах 7, 6, 4, 2 і 0 відповідно.

Команди LAHF і SAHF використовуються для забезпечення програмної сумісності 8-розрядних МП зі старшими моделями, а також для аналізу стану прапорців і їх зміни за необхідності.

Команди передавання даних з використанням стека (стекові команди) – це команди звернення до стека для завантаження або вивантаження даних. Головна особливість роботи зі стеком – одночасне завантаження/вивантаження двох байтів і автоматична модифікація вмісту

регістра вказівника стека SP. До виконання стекових команд необхідно ініціювати регістри SS (регістр сегмента стека) і SP (регістр – вказівник стека).

Команда **PUSH (PUSH operand onto stack)** – завантаження операнда у стек. Синтаксис команди:

PUSH *src*.

Як операнд *src* (джерело) у цій команді можуть бути регістри загального призначення, регістри вказівники, сегментні регістри, комірки пам'яті і безпосередні дані. Довжина операнда становить 2 або 4 байти.

Виконання команди полягає у завантаженні вмісту регістра/комірки пам'яті у дві комірки пам'яті з адресами, що визначаються вмістом регістра SP і становлять, для операндів довжиною 2 байти – (SP)–1 і (SP)–2, причому вміст старшого байта операнда записується у комірку з адресою (SP)–1, а молодшого – у комірку з адресою (SP)–2. Якщо використовуються 4-байтові операнди, значення вказівника стека становлять (SP)–1, (SP)–2, (SP)–3 і (SP)–4. Команду PUSH необхідно використовувати у парі з командою вивантаження POP.

Команда **POP (POP operand from the stack)** – вивантаження операнда зі стека. Синтаксис команди:

POP *dst*.

Команда завантажує в операнд *dst* (приймач) слово або подвійне слово зі стека. Як операнд *dst* (приймач) у цій команді можуть бути регістри загального призначення, регістри вказівники, сегментні регістри і комірки пам'яті. Довжина операнда становить 2 або 4 байти. Відповідно довжині операнда модифікується вміст регістра SP, який після виконання команди набуває значення (SP)+2 при довжині операнда 2 байти, і становить (SP)+4, якщо довжина операнда становить подвійне слово.

Команда **PUSHA (PUSH All general registers onto stack) –** завантаження вмісту всіх регістрів загального призначення у стек. Синтаксис команди:

PUSHA.

При виконанні команда розміщує у стеку вміст усіх регістрів загального призначення у наступному порядку: AX, CX, DX, BX, SP, BP, SI, DI. Вміст DI буде розташовано у новій вершині стека (у комірці пам'яті з найменшою адресою). У стеку буде зберігатися вміст регістра SP, що був до початку виконання цієї операції. Значення вказівника стека SP, після виконання команди, буде дорівнювати (SP)–16.

Для роботи з 32-розрядними регістрами загального призначення є команда **PUSHAD**.

Команда **POPA (POP All general registers from the stack)** – вивантаження вмісту всіх регістрів загального призначення у стек. Синтаксис команди:

POPA.

Команда вивантажує вміст всіх регістрів загального призначення у порядку, зворотному тому, який було описано у команді PUSHA. При цьому вміст регістра SP вивантажується, але не відновлюється. Значення вказівника стека SP після виконання команди збільшується на 16.

Для роботи з 32-розрядними регістрами загального призначення є команда **POPAD.**

Команда **PUSHF (PUSH Flags registers onto stack) –** завантаження вмісту регістра прапорців у стек. Синтаксис команди:

PUSHF.

Команда **POPF (POP Flags registers from the stack)** – вивантаження вмісту регістра прапорців у стек. Синтаксис команди:

POPF.

Команди PUSHF і POPF можуть використовуватися для звернення до регістра прапорців як до звичайного регістра у програмах обробки переривань і інших випадках, коли необхідно зберегти локальні ознаки обчислень.

Команди введення-виведення – це команди, за допомогою яких можливо обмінюватися даними з периферійними пристроями.

Існують дві команди:

– команда введення **IN (INput operand from port)**. Синтаксис команди:

IN *dst, src*,

де як операнд *dst* (приймач) використовується вміст акумулятора AL, AX або EAX, а як операнд *src* (джерело) – адреса (номер) порта у вигляді безпосереднього числа або як вміст регістра DX;

– команда виведення **OUT (OUT operand to port).** Синтаксис команди

OUT *dst, src*,

де операнд *dst* (приймач) – адреса (номер) порта у вигляді безпосереднього числа або як вміст регістра DX, а операнд *src* (джерело) – акумулятор AL, AX або EAX.

Якщо номер порта задається безпосередньо, то можливо адресувати до 255 портів а якщо задається вмістом регістра DX, то допускається формування і звернення до 65536 адрес різних портів. Непряме адресування через регістр DX зручно використовувати, якщо адреса порту обчислюється у ході виконання програми.

Команди не змінюють значення регістра прапорців.

Приклади команд введення- виведення:

```
IN AX,22H    ; Ведення у регістр AX слова з порту, номер якого 22H
IN AL, DX    ; Введення у регістр AL байта з порту, номер якого
             ; завантажено до регістра DX
OUT DX,AX    ; Виведення слова через порт, номер якого завантажено
             ; до регістра DX
```

Команди передавання рядків даних – це команди, що дозволяють обмінюватися даними у вигляді рядків байтів, слів або довгих слів. Рядок – це контекстно-зв'язана послідовність даних, що містяться в суміжних комірках пам'яті. Загалом у системі команд є п'ять операцій, кожна з яких зреалізована трьома командами.

Пересилання елементів рядок-джерело у рядок-приймач відбувається в залежності від довжини елемента рядка за допомогою команд:

MOVSB (MOVe String Byte) – пересилання байтів;

MOVSW (MOVe String Word) – пересилання слів;

MOVSD (MOVe String Double word) – пересилання подвійних слів.

Ці команди копіюють елемент рядок-джерело довжиною байт, слово або подвійне слово, що знаходиться за адресою DS:SI в елемент рядок-приймач, що визначається адресою ES:DI. Адреса операнда-приймача може визначатися тільки регістром ES, заміна адреси операнда-джерела дозволяється, за умовчанням визначено DS.

Після пересилання вміст регістрів SI і DI автоматично змінюється відповідно значення прапорця DF. Якщо прапорець DF дорівнює 0 (була виконана команда CLD), то вміст регістрів SI і DI збільшується, якщо DF має значення 1 (була виконана команда STD), то вміст регістрів SI і DI зменшується. Збільшення/зменшення відбувається на 1, якщо пересилається байт, на 2, якщо пересилається слово, і на 4, якщо пересилається подвійне слово.

Команді може передувати префікс REP, який примушує виконувати пересилання поки вміст регістра CX не стане дорівнювати 0.

Команди завантаження елемента з рядка в акумулятор AL/AX/EAX мають вигляд:

LODSB (LOad String Byte operands) – завантаження байта;

LODSW (LOad String Word operands) – завантаження слова;

LODSD (LOad String Double word operands) – завантаження подвійного слова.

Ці команди завантажують елемент рядок-джерело довжиною байт, слово або подвійне слово, що знаходиться за адресою DS(ES/GS/FS):SI в акумулятор. Після пересилання вміст регістра SI автоматично змінюється відповідно значення прапорця DF. Якщо прапорець DF дорівнює 0 (була виконана команда CLD), то вміст регістра SI збільшується, якщо DF має значення 1 (була виконана команда STD), то вміст регістра SI зменшується. Збільшення/зменшення відбувається на 1, якщо пересилається байт, на 2, якщо пересилається слово, і на 4, якщо пересилається подвійне слово.

Команді може передувати префікс REP, однак більш доцільним є використання цієї команди у LOOP-конструкціях, тому що зазвичай необхідне подальше оброблення кожного елементу.

Команди збереження вмісту акумулятора AL/AX/EAX у рядку мають вигляд:

STOSB (Store String Byte operands) – завантаження байта;

STOSW (Store String Word operands) – завантаження слова;

STOSD (Store String Double word operands) – завантаження подвійного слова.

Ці команди завантажують дані з акумулятора AL/AX/EAX довжиною байт, слово або подвійне слово у рядок, елемент якого знаходиться за адресою ES:DI. Після пересилання вміст регістра DI автоматично змінюється відповідно значення прапорця DF. Якщо прапорець DF дорівнює 0 (була виконана команда CLD), то вміст регістра DI збільшується, якщо DF має значення 1 (була виконана команда STD), то вміст регістра DI зменшується. Збільшення/зменшення відбувається на 1, якщо пересилається байт, на 2, якщо пересилається слово, і на 4, якщо пересилається подвійне слово.

Команді може передувати префікс REP, що дає можливість формувати блоки пам’яті, кількість елементів у яких буде визначатися вмістом регістра CX.

Команди введення рядка даних через порт мають вигляд:

INSB (INput String Byte operands) – введення рядка байт з порту;

INSW (INput String Word operands) – введення рядка слів з порту;

INSD (INput String Double word operands) – введення рядка подвійних слів з порту.

Ці команди завантажують у пам’ять, адреса якої визначається вмістом регістрів ES:DI, байт, слово або подвійне слово з порту, номер якого задано у регістрі DX. Використовувати у цій команді інші регістри для адресування не дозволяється. Після пересилання вміст регістра DI автоматично змінюється відповідно значення прапорця DF. Якщо прапорець DF дорівнює 0 (була виконана команда CLD), то вміст регістра DI збільшується, якщо DF має значення 1 (була виконана команда STD), то вміст регістра DI зменшується. Збільшення/зменшення відбувається на 1, якщо пересилається байт, на 2, якщо пересилається слово і на 4, якщо пересилається подвійне слово.

Команді може передувати префікс REP, що дає можливість приймати рядки, кількість елементів у яких буде визначатися вмістом регістра CX.

Команди виведення рядка даних через порт мають вигляд:

OUTSB (INput String Byte operands) – виведення рядка байт через порт;

OUTSW (INput String Word operands) – виведення рядка слів через порт;

OUTSD (INput String Double word operands) – виведення рядка подвійних слів через порт.

Ці команди виводять через порт, номер якого задано вмістом регістра DX, дані з пам'яті, яку адресовано регістрами ES:SI, байт, слово або подвійне слово. Дозволяється заміна сегментного регістра для адресування пам'яті. Номер порта може задаватися лише регістром DX. Після пересилання вміст регістра SI автоматично змінюється відповідно до значення прапорця DF. Якщо прапорець DF дорівнює 0 (була виконана команда CLD), то вміст регістра SI збільшується, якщо DF має значення 1 (була виконана команда STD), то вміст регістра SI зменшується. Збільшення/зменшення відбувається на 1, якщо пересилається байт, на 2, якщо пересилається слово, і на 4, якщо пересилається подвійне слово.

Команді може передувати префікс REP, що дає можливість виводити рядки, кількість елементів у яких буде визначатися вмістом регістра CX.

Команди перетворення типів даних використовуються для формування даних однієї розмірності (кількості розрядів) перед виконанням арифметичних операцій. Особливе значення ці команди набувають, якщо в арифметичних командах використовуються числа зі знаком. Їх виконання ґрунтується на поширенні знакового біта на всі старші розряди числа. Таким чином, формуються числа з тим самим значенням і знаком, але з більшою кількістю розрядів. Команди цієї групи виконуються з фіксованими регістрами, тому операндів у них немає. Усі команди цієї групи не змінюють значення регістра прапорців.

Команда **CBW (Convert Byte to Word)** – перетворення байта (у регістрі AL) у слово (у регістрі AX). При виконанні знаковий розряд числа з регістра AL поширюється на всі розряди регістра AH. Наприклад, якщо до виконання команди вміст регістра AL у регістрі AX був 85H (число –05 у доповнювальному коді), то після її виконання вміст регістра AX буде FF85H. Якщо вміст регістра AL був 74H (число +74H), то після виконання команди вміст регістра AX стане 0075H.

Команда **CWD (Convert Word to Double Word)** – перетворення слова (у регістрі AX) у подвійне слово (у регістрах DX:AX). Виконання команди полягає у поширенні знакового розряду з регістра AX на усі розряди регістра DX.

Операцію можливо використовувати при підготовці до виконання операції ділення, в якій розмір діленого повинен бути у два рази більший ніж розмір дільника. Наприклад, якщо необхідно у програмі поділити число, що знаходиться у регістрі AX на число з регістра BX. Використання цієї команди показано у прикладі виконання команди DIV.

Команда **CWDE (Convert Word to Double Word Extended)** – перетворення слова (у регістрі AX) у подвійне слово (у регістрі EAX). При виконанні знаковий розряд числа з регістра AX поширюється на всі старші розряди регістра EAX.

До цієї групи також можливо віднести команди MOVSX і MOVZX, які були розглянуті раніше.

Контрольні запитання:

1 Які типи команд входять до групи команд пересилань?

2 Чим відрізняється використання команд MOV, MOVSX і MOVZX?

3 Покажіть вміст регістрів DS і BX після виконання команди LDS BX, [1234], якщо комірки пам’яті у поточному сегменті мають вміст: 7000:1234 10 20 30 40 50?

4 Які команди введення-виведення Ви знаєте?

5 Напишіть коментарі з поясненням виконання кожної команди до наступної програми:

```
MOV CX,0004H ; Завантаження регістра-лічильника
CLD
MOV SI,1234H
REP LODSB
```

Назвіть вміст регістрів SI і AL після виконання цієї програми, якщо вміст комірок пам’яті у поточному сегменті був такий самий, як у запитанні 3.

6 Для чого використовуються команди перетворення типів даних?

Контрольні запитання підвищеної складності:

1 Яким чином виконуєтьсяя перетворення типів даних зі знаком?

2 Для чого використовується префікс REP?

3 Наведіть вміст регістра – вказівника стека SP і вміст комірок пам’яті з адресами 3456H і 3455H після виконання команди PUSH CX, якщо до виконання команди регістри мали вміст: (SP) = 3457H і (CX) = 1234H?

9.3.3 Команди перетворення даних мови Асемблер-86

Вхідний контроль:

1 Які математичні та логічні операції виконуються в обчислювальній техніці?

2 Який цифровий пристрій виконує математичні та логічні операції?

3 Які ознаки результату формуються пристроєм АЛП при виконанні математичних та логічних операцій?

4 У яких системах числення можуть бути подані дані, які оброблюються МПС?

5 Як подаються десяткові числа у двійково-десятковій кодованій системі числення?

6 Як формуються ознаки результатів при виконанні арифметичних і логічних операцій?

До цієї групи можливо віднести арифметичні і логічні команди, а також команди, які у своїй роботі використовують арифметичні або логічні принципи виконання.

Арифметичні команди мови Асемблер-86 виконуються над цілими числами чотирьох типів: двійкові числа без знака; двійкові числа зі знаком; розпаковані двійково-десяткові числа й упаковані двійково-десяткові числа. Числа зі знаком подаються у МП у вигляді доповняльного коду. Для виконання арифметичних операцій над числами з плавучою точкою використовується математичний співпроцесор, команди якого становлять окрему групу команд.

Команди двійкової арифметики

До групи арифметичних команд над двійковими числами входять команди: додавання, віднімання, множення, ділення і зміни знака.

Команди додавання представлені трьома командами:

ADD (ADDition) – команда додавання операндів. Команда має вид

ADD *dst*, *src*

і виконує додавання операндів *src* (джерело) і *dst* (приймач), які можуть бути байтами, словами, подвійними словами. Результат операції розміщується в операнді *dst* (приймач), при цьому втрачається один з доданків. Як операнд *dst* (приймач) можуть використовуватися усі регістри загального призначення, крім сегментних, і комірки пам’яті, які можливо адресувати будь-яким способом адресування. У будь-якому випадку адресується лише одна комірка, що відповідає молодшому байту результату, адреси інших для розміщення старших байтів формуються автоматично. В разі використання як операнд *dst* (приймач) акумулятора, довжина команди стає меншою і виконується вона швидше.

Операнд *src* (джерело) може розміщуватись у регістрах загального призначення, комірках пам’яті і бути безпосереднім числом. Розмір обох операндів обов’язково повинен співпадати.

Команда установлює прапорці OF, SF, ZF, AF, PF, CF регістра ознак (FLAGS), які можливо аналізувати у подальших командах.

ADC (Addition with Carry) – команда додавання операндів з урахуванням вхідного перенесення з біта CF регістра ознак, значення якого додається до результату.

Вид і виконання команди повністю співпадає з командою ADD.

Використовується при додаванні двійкових 64-розрядних чисел.

Приклади використання команд додавання:

```
ADD AX, SI          ; Додавання вмісту регістра AX до вмісту SI і
                    ; збереження результату у регістрі AX
ADD [1234],BX       ; Додавання вмісту комірок пам'яті з адресами
                    ; 1234H і 1235H у поточному сегменті даних до
                    ; вмісту регістра BX і завантаження результату до
                    ; тих самих комірок. Наприклад, вміст регістра
                    ; (BX) = 1111H; у комірках пам'яті з адресами
                    ; 1234H і 1235H зберігаються відповідно числа 22H
                    ; і 33H. Додавання буде виконуватися таким чином
                    ;       (BX)        1111H
                    ;       [1234]      0022H
                    ;       [1235]      3300H
                    ;                   4433H
                    ; Результат буде розташовано у тих самих комірках –
                    ; молодший байт у комірці з адресою 1234H, а
                    ; старший – у комірці 1235H
ADC AL,DH           ; Додавання до вмісту регістра DH вмісту AL і
                    ; поточного значення біту CF, завантаження
                    ; результату до регістра AL
                    ;       (AL)        1FH
                    ;       (DH)        04H
                    ;       CF          01
                    ;                   24H
```

INC (INCrement operand by 1) – команда додавання 1 до значення операнда. Синтаксис команди:

INC *dst*.

Операндом *dst* (приймач) може бути будь-який регістр загального призначення (8- , 16- або 32-розрядний) або операнд (8- , 16- або 32-розрядний), розташований у пам'яті. При виконанні команди операнд вважається беззнаковим числом.

Команда установлює прапорці OF, SF, ZF, AF, PF регістра ознак.

Команди віднімання також представлені трьома командами:
SUB (SUBtract) – команда віднімання цілих чисел. Команда має вид

SUB *dst*, *src*

і виконує віднімання від вмісту операнда *dst* (приймач) *src* (джерело) вмісту операнда *src* (джерело). Результат зберігається в операнді *dst* (приймач), при цьому втрачається попередній вміст цього операнда. Як операнди *dst* (приймач) можуть використовуватися усі регістри загального призначення, крім сегментних, і комірки пам'яті, які можливо адресувати будь-яким способом адресування. У будь-якому випадку адресується лише одна комірка, що відповідає молодшому байту результату, адреси інших для розміщення старших байтів формуються автоматично. В разі використання як операнд *dst* (приймач) акумулятора, довжина команди стає меншою і виконується вона швидше.

Операнд *src* (джерело) може розміщуватись у регістрах загального призначення, комірках пам'яті і бути безпосереднім числом. Розмір обох операндів обов'язково повинен співпадати.

Команда змінює прапорці OF, SF, ZF, AF, PF, CF регістра ознак, які можливо аналізувати у подальших командах.

SBB (SuBtract with Borrow) – віднімання з урахуванням позики зі старшого розряду. Використовується для виконання віднімання старших частин значень багатобайтових операндів з урахуванням можливої попередньої позики під час віднімання молодших частин операндів.

При її виконанні, від значення операнда *dst* (приймач) віднімається сума значення операнда *src* (джерело) і значення біта CF; результат завантажується на місце операнда *dst* (приймач). Вид команди співпадає з командою SUB.

Команда змінює прапорці OF, SF, ZF, AF, PF, CF регістра ознак.

Приклад використання цієї команди покажемо за допомогою наступної програми:

```
STC              ; Установлення біта CF в 1
MOV AX,4567H  ; Завантаження до регістра AX числа 4567H
MOV BX,1111H  ; Завантаження до регістра BX числа 1111H
SBB AX,BX       ; Виконання цієї команди відбувається у два етапи:
                 ; – до вмісту регістра BX додається 1
                 ;          1111H + 0001H = 1112H;
                 ; – із вмісту регістра AX віднімається число, яке
                 ; отримано на попередньому етапі:
                 ;          _ 4567H
                 ;            1112H
                 ;            3455H
                 ; при виконанні прапорець CF скидається в 0
                 ; (позики немає)
MOV CX,1234H  ; Завантаження до регістра CX числа 1234H
```

```
MOV DX,9111H  ; Завантаження до регістра DX числа 9111H
SBB CX,DX     ; Виконання команди відбувається аналогічно
              ; описаному вище:
              ;       9111H + 0000H = 9111H (прапорець CF
              ;   скинуто);
              ;          _ 1234H
              ;            9111H
              ;            8123H   (з урахуванням позики зі
              ;     старшого розряду; прапорець CF
              ;     установлюється в 1)
```

DEC (DECrement operand by 1) – віднімання від значення операнда 1. Синтаксис команди:

DEC *dst*.

Операндом *dst* (приймач) може бути будь-який регістр загального призначення (8-, 16- або 32-розрядний) або операнд (8-, 16- або 32-розрядний), розташований у пам'яті. При виконанні команди операнд вважається беззнаковим числом.

Команда змінює прапорці OF, SF, ZF, AF, PF регістра ознак.

Команди множення у системі команд представлені двома командами: для цілих беззнакових чисел і для цілих чисел зі знаком.

MUL (MULtiply) – команда множення двох цілих беззнакових чисел. Команда має узагальнений вид

MUL *src*.

Алгоритм виконання залежить від формату операнда у команді і потребує визначити лише операнд *src* (джерело), розмір якого байт, слово або подвійне слово і який може розміщатися у пам'яті або у регістрі. Розміщення другого операнда фіксовано і залежить від розміру операнда *src* (джерело):

– якщо операнд *src* (джерело) – байт, то другий операнд повинен розміщуватися у регістрі AL;

– якщо операнд *src* (джерело) – слово, то другий операнд повинен розміщуватися у регістрі AX;

– якщо операнд *src* (джерело) – подвійне слово, то другий операнд повинен розміщуватися у регістрі EAX.

Розміщення результату також є фіксованим і визначається розмірами співмножників:

– якщо перемножуються байти, то результат розміщується у регістрі AX;

– якщо перемножуються слова, то результат розміщується у регістрах DX:AX;

– якщо перемножуються подвійні слова, то результат розміщується у регістрах EDX:EAX.

У результаті виконання цієї команди формуються лише прапорці OF і CF. Їх значення, якщо старша половина добутку не дорівнює нулю, становить 1, що показує наявність розрядів результату у регістрах AH і DX. В іншому випадку їх значення дорівнює 0. На стан інших прапорців виконання команди MUL не впливає.

Приклад використання команди:

```
MOV AL,02H  ; Завантаження першого співмножника (02H = 02D)
            ; до регістра AL
MOV BL,80H  ; Завантаження другого співмножника (80H = 128D)
            ; до регістра BL
MUL BL      ; Множення операндів, результат розташовано у
            ; регістрі AX (0100H = 256D)
```

IMUL (Integer MULtiply) – команда множення двох цілих чисел зі знаком. Команда має узагальнений вид

IMUL *src*.

Команда виконується майже так само, як і команда MUL, проте у цій команді операнди подаються як числа зі знаком у доповнювальному коді.

Прапорці OF і CF установлюються залежно від розміщення знакових розрядів: якщо старша половина добутку, яка знаходиться у регістрах AH або DX, не містить поширення знаку молодшої частини результату, то прапорці OF і CF установлюються в 1, що показує наявність у старшій половині результату значущих цифр. В іншому випадку – установлюються у 0. Стан інших прапорців є невизначеним.

Команди ділення у системі команд представлено двома командами: для цілих беззнакових чисел і для цілих чисел зі знаком.

DIV (DIVide unsigned) – команда ділення цілих беззнакових чисел. Узагальнена форма цієї команди має вигляд:

DIV *src*.

Для виконання команди необхідно задавати лише один операнд – *src* (джерело), розмір якого може становити байт, слово і подвійне слово, розміщення другого операнду є фіксоване і залежить від розміру дільника, який визначено у команді:

– якщо дільник це байт, то ділене повинне бути розміщене у регістрі AX. Частку буде розташовано у регістрі AL, а залишок – у регістрі AH;

– якщо дільник це слово, то ділене розміщується у регістрах DX:AX і старша частина діленого розміщується у DX. Частка розміщується в AX, а залишок у регістрі DX;

– якщо дільник це подвійне слово, то ділене розміщується у регістрах EDX:EAX і старша частина діленого розміщується у DX. Частка розміщується в EAX, а залишок – у регістрі EDX.

Якщо при діленні формується дробна частка, то вона заокруглюється до цілого числа у результаті відкидання дробної частини.

Виконання команди не впливає на прапорці регістра ознак.

Якщо частка за розміром більша за акумулятор або дільник дорівнює 0, то генерується переривання типу 0 і програма виконує підпрограму переривання.

Приклад виконання такої команди:

```
MOV AX,1111H  ; Завантаження діленого (1111H = 4369D) до
              ; регістра AX
MOV BX,20H    ; Завантаження дільника (20H = 32D) до регістра BL
CWD           ; Формування необхідного розміру діленого
DIV BX        ; Ділення, частка у регістрі AX дорівнює 0088H,
              ; залишок у регістрі DX дорівнює 0011H
```

IDIV (Integer DIVide) – команда ділення цілих чисел зі знаком. Узагальнена форма цієї команди має вигляд:

IDIV *src*.

Команда виконується майже так само як і команда DIV, проте у цій команді операнди подаються як числа зі знаком у доповнювальному коді.

Стан прапорців у регістрі ознак є невизначеним.

NEG (NEGate operand) – команда, яка змінює знак операнда. Команда має вигляд:

NEG *src*.

Команда змінює знак операнда, який може бути вмістом регістра загального призначення (8-, 16- або 32-розрядним) або вмістом комірок пам’яті (8, 16 або 32 розряди). Операція двійкового доповнення виконується як інвертування всіх розрядів операнда і додавання до результату 1. Команда змінює лише прапорець CF.

Команди десяткової арифметики

До цієї групи входять команди, що дозволяють працювати з десятковими числами, які можуть бути поданими у двох формах: як упаковані двійково-десяткові числа (BCD-формат), і як розпаковані двійково-десяткові числа (ASCII-формат). У кожному з форматів багаторозрядні десяткові числа представлено послідовностями байтів, з якими і працює МП. Основним робочим регістром в усіх командах десяткової арифметики є регістр AL. Слід зазначити, що команд, спеціально налаштованих на виконання операцій складання, віднімання, множення і ділення десяткових чисел немає, тому що розмірність таких чисел може бути довільною, крім того, виконання множення і ділення десяткових чисел є можливе лише в розпакованому форматі. Тому виконання арифметичних операцій виконується за допомогою команд двійкової арифметики, а потім проводиться корекція результату.

Корекція результату додавання десяткових чисел виконується після виконання команди ADD або ADC. Як було показано раніше, неправильний результат може бути обумовлений двома причинами:

– у результаті додавання отримали тетраду, двійковий еквівалент якої є більший ніж 9;

– отримано перенесення до старшої тетради з вагою 16.

Корекція результату проводиться за результатами аналізу прапорців AF і CF регістра ознак, які визначаються вмістом регістра AL, в якому розміщено суму.

DAA (Decimal Adjust for Addition) – команда десяткової корекції додавання BCD-чисел, які подано в упакованому виді.

Операндів у команди немає, тому що вона працює лише з регістром AL. Алгоритм корекції результату виконання попередньої команди додавання виконується за таким алгоритмом:

– якщо прапорець AF = 1 або молодша тетрада регістра AL має значення, більше ніж 9, то до вмісту AL додається число 06;

– якщо прапорець CF = 1 або старша тетрада регістра AL має значення, більше ніж 9, то до вмісту AL додається число 60;

– якщо мають місце обидві попередні ситуації, то корекція починається з молодшого розряду.

Після виконання команди DAA необхідно перевіряти значення прапорця CF для його урахування при роботі зі старшими розрядами десяткових цифр числа.

AAA (Ascii Adjust for Addition) – команда десяткової корекції додавання BCD-чисел, які подано у розпакованому виді. Команда також працює лише з регістром AL. Алгоритм роботи:

– якщо молодша тетрада регістра AL є припустима і значення AF = 0, то необхідно обнулити старшу тетраду AL;

– якщо молодша тетрада регістра AL більша 9 або значення AF = 1, то необхідно додати 06 до вмісту регістра AL і додати 1 до вмісту регістра AH;
– обнулити старшу тетраду регістра AL;
– установити прапорець CF у такий стан, в якому знаходиться AF.

Корекція результату віднімання десяткових чисел виконується після виконання команди SUB або SBB. Корекція результату проводиться за результатами аналізу прапорців AF і CF регістра ознак, які визначаються вмістом регістра AL, в якому розміщено віднімання.

DAS (Decimal Adjust for Subtraction) – команда десяткової корекції після віднімання. Алгоритм роботи команди:
– якщо прапорець AF = 1 або молодша тетрада AL має вміст, більший ніж 9, то з вмісту AL віднімається 06;
– якщо CF = 1 або старша тетрада AL має вміст, більший ніж 9, то з вмісту AL віднімається 60.

AAS (Ascii Adjust after Subtraction) – команда десяткової корекції після віднімання BCD-чисел, які подано у розпакованому виді. Алгоритм роботи команди:
– якщо молодша тетрада регістра AL припустима і AF = 0, то обнулити старшу тетраду регістра AL;
– якщо вміст молодшої тетради регістра AL не є припустимий або AF = 1, то необхідно відняти число 06 зі вмісту AL;
– обнулити старшу тетраду регістра AL;
– надати CF такий же стан як і CF.

Корекція результатів множення виконується командою
AAM (Ascii Adjust after Multiply) – команда корекції результату множення двох розпакованих BCD-чисел і перетворення результату у вигляді двійкового числа, меншого 63H (99D), у його розпакований BCD-еквівалент.

Команда здійснює ділення вмісту регістра AL на десять (0AH) і завантажує частку у регістр AH, а залишок – у регістр AL. Стан прапорців SF, ZF, PF визначається вмістом AL, а стан прапорців OF AF CF не є визначений.

Корекція результатів ділення проводиться до виконання ділення командою
AAD (Ascii Adjust before Division) – команда підготовки двох розпакованих BCD-чисел для ділення і перетворення двозначного розпакованого числа, меншого 63H (99D), у двійковий еквівалент. Алгоритм роботи команди:
– вміст регістра AH множиться на 10D і додається до вмісту регістра AL;
– регістр AH обнулюється.

Виконується ділення числа у регістрі AX за допомогою команди DIV.

До групи арифметичних команд також входять такі команди:

BSWAP (Byte SWAP) – команда перестановки байтів. Синтаксис команди

BSWAP *src*.

Операндом *src* (джерело) може бути лише 32-розрядний регістр загального призначення. Команда використовується для зміни порядку слідування байтів і переходу від однієї форми адресування до іншої (від форми «молодший байт за молодшою адресою» на звернену). Команда не змінює вміст регістра прапорців. Алгоритм роботи можливо пояснити за допомогою рис. 9.11.

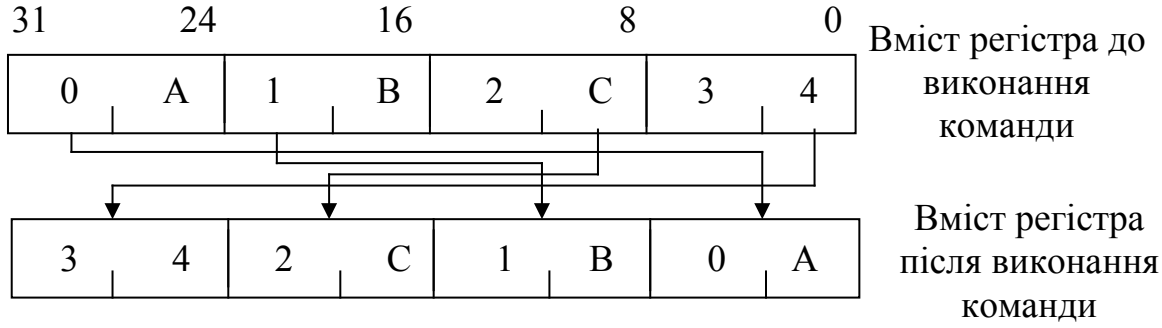


Рисунок 9.11 – Алгоритм виконання команди BSWAP

XADD (eXchange and ADD) – обмін і додавання значень двох операндів. Команда має вигляд

XADD *dst*, *src*.

Операнд *dst* (приймач) може бути 8-, 16- або 32-розрядним регістром загального призначення або коміркою пам’яті з такою ж кількістю розрядів. Операнд *src* (джерело) може бути лише пам’яттю відповідної розмірності.

Команда виконується у три етапи:

– вміст операнда *dst* (приймач) копіюється в операнд *src* (джерело);

– виконується додавання вмісту операнда *dst* (приймач) до вмісту операнда *src* (джерело);

– сума завантажується в операнд *dst* (приймач). Команда змінює усі прапорці результату.

CMP (CoMPare operands) – команда для порівняння двох операндів. Команда має вигляд

CMP *dst*, *src*.

Операнд *dst* (приймач) розміщується у регістрі загального призначення або комірці пам’яті, а операнд *src* (джерело) також може бути вмістом регістра загального призначення або комірки пам’яті, але одночасно не можуть бути вмістом комірок пам’яті. Крім того, операндом *src* (джерело) може бути безпосереднє число.

Алгоритм виконання команди полягає у відніманні від вмісту операнда *dst* (приймач) вмісту операнда *src* (джерело). Результат не формується, лише установлюються ознаки результату, які можливо аналізувати і робити висновки про співвідношення операндів між собою.

Порівняння елементів рядків байтів, слів, подвійних слів відбувається в залежності від довжини елемента рядків за допомогою команд:

CMPS (CoMPare String) – команда для порівняння елементів двох рядків;

CMPSB (CoMPare String Byte) – команда для порівняння двох рядків байтів;

CMPSW (CoMPare String Word) – команда для порівняння двох рядків слів;

CMPSD (CoMPare String Double word operands) – команда для порівняння двох рядків подвійних слів;

Команда порівняння рядків має узагальнений вид

CMPS *dst*, *src*.

Команди CMPSB, CMPSW, CMPSD опереандів не мають.

Ці команди порівнюють елементи рядка-джерела довжиною байт, слово або подвійне слово, що знаходиться за адресою DS:SI з елементами рядка-приймача, адреса яких визначаються вмістом регістра ES:DI. Зміна регістрів, які визначають адреси оперендів не дозволена.

Алгоритм виконання команди полягає у відніманні від вмісту операнда *dst* (приймач) вмісту операнда *src* (джерело). Результат не формується, лише установлюються ознаки результату, які можливо аналізувати і робити висновки про співвідношення операндів між собою.

Після віднімання вміст регістрів SI і DI автоматично змінюється відповідно значення прапорця DF. Якщо прапорець DF дорівнює 0 (була виконана команда CLD), то вміст регістрів SI і DI збільшується, якщо DF має значення 1 (була виконана команда STD), то вміст регістрів SI і DI зменшується. Збільшення/зменшення відбувається на 1, якщо пересилається байт, на 2, якщо пересилається слово, і на 4, якщо пересилається подвійне слово.

Команді може передувати префікс REP, який примушує виконувати порівняння, до поки вміст регістра CX не стане дорівнювати 0.

CMPXCHG (CoMPare and eXCHanGe) – команда для порівняння двох операндів і обміну даними між ними. Синтаксис команди

CMPXCHG *dst*, *src*.

Операнд *dst* (приймач) розміщується у регістрі загального призначення або комірці пам'яті, а операнд *src* (джерело) також може бути вмістом регістра загального призначення або комірки пам'яті, але одночасно не можуть бути вмістом комірок пам'яті.

Команда порівнює значення операндів, установлює значення прапорця ZF = 1, якщо операнди рівні або ZF = 0 в іншому випадку, після чого обмінює вміст операндів між собою.

SETcc (byte SET on condition) – команда установлення вмісту байта при виконанні певної умови. Команда має вигляд

SETcc *src*.

Операнд у цій команді 8-розрядний регістр або комірка пам'яті.

Команда виконує установлення вмісту операнда 01H, якщо умова виконується, або у 00H, якщо не виконується.

Значення модифікатора кода операції cc аналогічно значенню модифікатора команд умовних переходів, де вони будуть розглянуті.

Сканування елементів рядка байтів, слів, подвійних слів відбувається в залежності від довжини елемента рядка за допомогою команд:

SCAS (SCAn String) – команда для сканування елементів рядка;

SCASB (SCAn String Byte) – команда для сканування елементів рядка байтів;

SCASW (SCAn String Word) – команда для сканування елементів рядка слів;

SCASD (SCAn String Double word) – команда для сканування елементів рядка подвійних слів;

Команда сканування елементів рядка має узагальнений вид

SCAS *dst*.

Команди CMPSB, CMPSW, CMPSD опереандів не мають.

Алгоритм виконання команди полягає у відніманні елемента рядка – операнда *dst* (приймач), який адресується тільки регістром ES:DI, від вмісту акумулятора Al (байт), AX (слово), EAX (подвійне слово). Результат не формується, лише установлюються ознаки результату, які можливо аналізувати і робити висновки про співвідношення операндів між собою.

Після віднімання вміст регістра DI автоматично змінюється відповідно значення прапорця DF. Якщо прапорець DF дорівнює 0 (була виконана команда

CLD), то вміст регістру DI збільшується, якщо DF має значення 1 (була виконана команда STD), то зменшується. Збільшення/зменшення відбувається на 1, якщо пересилається байт, на 2, якщо пересилається слово, і на 4, якщо пересилається подвійне слово.

Команді може передувати префікс REP, який примушує виконувати порівняння поки вміст регістра CX не стане дорівнювати 0.

Команди логічних операцій

До цієї групи команд відносяться команди, які при роботі використовують правила алгебри логіки. До таких команд відносяться команди, яки виконують логічні операції над операндами, команди обробки біт і команди зсувів.

Логічні операції мови Асемблер-86:

AND (logical AND) – команда логічного множення операндів (кон'юнкція).

OR (logical OR) – команда логічного додавання операндів (диз'юнкція).

XOR (logical eXclusive OR) – логічне виключне АБО.

TEST (TEST operand) – логічне ТА. Використовується для логічного порівняння операндів.

NOT (NOT operand) – інвертування операнда.

Узагальнене представлення логічних команд має вигляд:

AND *dst*, *src*.
OR *dst*, *src*.
XOR *dst*, *src*.
TEST *dst*, *src*.
NOT *src*.

В усіх командах операндами можуть бути 8-, 16-, 32-розрядні регістри загального призначення і пам'ять з відповідною кількістю комірок. Операнд *src* (джерело) також може бути представлено безпосереднім числом (крім команди NOT).

Команди AND, OR, XOR, TEST змінюють прапорці таким чином:

OF і CF – завжди установлюються у нульовий стан;

SF, ZF, PF – установлюються відповідно до результату за тими самими правилами, що і для арифметичних операцій;

AF – не визначається.

Команди зсувів поділяються на арифметичні і логічні. Для логічних зсувів характерно, що біт, який виходить за межі регістра втрачається, а на місто біта, який зсунувся, у регістр записується 0. При виконанні арифметичних зсувів праворуч знаковий біт не втрачається, а зберігається у сусідньому, тим самим зберігаючи знак числа.

Асемблер включає такі команди зсувів:

SHL (Shift logical Left) – зсунути логічно ліворуч;

SAL (Shift Arithmetic operand Left) – зсунути арифметично ліворуч;

SHR (Shift logical Right) – зсунути логічно праворуч;

SAR (Shift Arithmetic operand Right) – зсунути арифметично праворуч;

RCL (Rotate operand through Carry flag Left) – зсунути циклічно ліворуч через перенесення;

RCR (Rotate operand through Carry flag Right) – зсунути циклічно праворуч через перенесення;

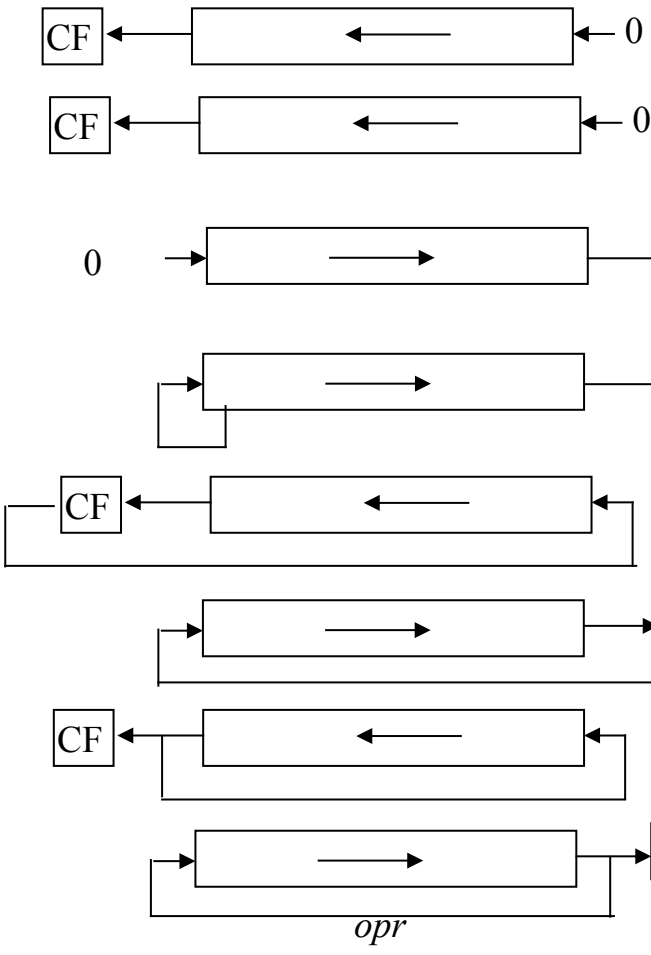
ROL (Rotate operand Left) – зсунути циклічно ліворуч;

ROR (Rotate operand Right) – зсунути циклічно праворуч.

Пояснення алгоритму роботи і синтаксис кожної з команд зсувів подано у табл. 9.3.

У таблиці прийнято умовні позначення: *opr* – операнд (регістр або комірка пам'яті, вміст яких зсувається); *cnt*– кількість зсувів (може приймати значення 1 або задаватися вмістом регістра-лічильника CL).

Таблиця 9.3 – Команди зсувів

Мнемоніка та формат команди	Опис виконання
SHL *opr,cnt*	
SAL *opr,cnt*	
SHR *opr,cnt*	
SAR *opr,cnt*	
RCL *opr,cnt*	
RCR *opr,cnt*	
ROL *opr,cnt*	
ROR *opr,cnt*	

Контрольні запитання:

1 Які команди додавання операндів у мові Асемблер Ви знаєте? Чим вони відрізняються одна від одної?

2 Чим різняться команди SUB і SBB? Поясніть на прикладі.

3 Як виконується команда DEC CX?

4 Як виконується команда MUL і в яких регістрах може розміщуватися операнд *src* (джерело) і операнд *dst* (приймач)?

5 Як виконується команда DIV і в яких регістрах може розміщуватися операнд *src* (джерело) і операнд *dst* (приймач)?

6 Для чого у командах ділення використовується команда CWD?

7 Для чого використовується команда CMP? Де розміщується результат виконання цієї команди?

8 Які команди логічних операцій Ви знаєте?

9 Який результат буде отримано при виконанні фрагмента програми:

```
MOV AX,AFH
SHL AX,1
```

10 Який результат буде отримано при виконанні фрагмента програми:

```
MOV AX,AFH
ROR AX,1
```

11 Який результат буде отримано при виконанні фрагмента програми:

```
MOV AX,AFH
RCL AX,1
```

Контрольні запитання підвищеної складності:

1 Чому при виконанні арифметичних команд над двійково-десятковими числами виникає потреба у корекції результату?

2 Як виконується корекція результату при виконанні арифметичних команд над двійково-десятковими числами?

3 Які прапорці формуються при виконанні логічних операцій?

4 Які команди зсувів можливо використовувати в якості команд множення і ділення на 2?

9.3.4 Команди умовних та безумовних переходів

Вхідний контроль:

1 Які ознаки результату, що формуються у МПС фірми Intel, Ви знаєте?

2 В якому випадку формується ознака ZF?

3 Яким чином формується ознака знаку результату SF?

4 Чим відрізняється формування ознак перенесення CF і переповнення OF?

5 Наведіть приклад реалізації умовного переходу будь-якою мовою високого рівня.

6 Наведіть приклад реалізації безумовного переходу будь-якою мовою високого рівня.

До групи таких команд відносяться команди, які можуть зробити перехід не до наступної за даною командою, а до команди, яка знаходиться в іншій комірці програмної пам’яті, адреса якої визначається адресою переходу.

Команди передачі керування підрозділяються на команди безумовних переходів, умовних переходів, викликів і повернення з підпрограм, команди переривань.

Команди безумовних переходів

При виконанні таких команд виконується зміна вмісту регістра PC або одночасно PC і CS. Команди такого типу мають довжину 2–5 байтів у залежності від типу переходу і довжини зміщення у команді, яким задається адреса переходу.

JMP (JuMP) – команда безумовного переходу

Команда має вигляд

JMP *мітка*.

Мітка може відповідати 8-, 16- або 32-розрядному зміщенню відносно наступної за JMP команди. При цьому змінюється лише вміст EIP/IP. Якщо мітка у команді – символічний ідентифікатор комірки пам'яті (16-, 32- або 48-розрядна адреса, то Асемблер визначає його як адресу, по якій необхідно зробити перехід. Така адреса може відповідати як близькому, так і далекому переходу.

Команди умовних переходів

При виконанні цих команд мова Асемблера робить аналіз певних ознак результатів і за результатами аналізу здійснює або не здійснює передачу керування.

Jcc (Jump if condition) – команда умовного переходу.

Перелік команд умовного переходу й умов, які перевіряються, подано у табл. 9.4.

Таблиця 9.4 – Команди умовних переходів

№ пп.	Дія	Мнемоніка та формат	Альтернативна мнемоніка	Умова, що перевіряється
1	Перейти, якщо нуль або дорівнює	JZ ADDR	JE ADDR	ZF=1
2	Перейти, якщо не нуль або не дорівнює	JNZ ADDR	JNE ADDR	ZF=0
3	Перейти, якщо знак установлено	JS ADDR		SF=1
4	Перейти, якщо знак скинуто	JNS ADDR		SF=0
5	Перейти, якщо є переповнення	JO ADDR		OF=1

Продовження табл.9.4

№ п.п	Дія	Мнемоніка та формат	Альтернативна мнемоніка	Умова, що перевіряється
6	Перейти, якщо немає переповнення	JNO ADDR		OF=0
7	Перейти, якщо паритет установлений	JP ADDR	JPE ADDR	PF=1
8	Перейти, якщо паритет скинуто	JNP ADDR	JPO ADDR	PF=0
9	Перейти, якщо вище (без знака)	JA ADDR		CF=0, ZF=0
10	Перейти, якщо нижче (без знака)	JNA ADDR		CF=1, ZF=1
11	Перейти, якщо вище/ або дорівнює (без знака)	JAE ADDR		CF=0
12	Перейти, якщо нижче/ або дорівнює (без знака)	JBE ADDR		CF=1, ZF=1
13	Перейти, якщо нижче/не вище або дорівнює (без знака)	JB ADDR	JNAE ADDR JC ADDR	CF=1
14	Перейти, якщо не нижче/ вище або дорівнює (без знака)	JNBE ADDR	JNB ADDR JNC ADDR	CF=0
15	Перейти, якщо є перенесення/позика	JC ADDR		CF=1
16	Перейти, якщо перенесення/позики немає	JNC ADDR		CF=0
17	Перейти, якщо менше/ не більше або дорівнює (зі знаком)	JL ADDR	JNGE ADDR	((CF) XOR(OF))=1
18	Перейти, якщо не менше/ більше або дорівнює (зі знаком)	JNL ADDR	JGE ADDR	((CF) XOR(OF))=0

Закінчення табл.9.4

№ п.п	Дія	Мнемоніка та формат	Альтернативна мнемоніка	Умова, що перевіряється
19	Перейти, якщо менше або дорівнює/не більше (зі знаком)	JLE ADDR		((SF) XOR(OF)OR(ZF))=1
20	Перейти, якщо не менше або дорівнює/більше (зі знаком)	JNLE ADDR	JG ADDR	((SF) XOR(OF)OR(ZF))=0

Використання і пояснення роботи деяких команд наведено у наступній програмі:

```
M1:     NOP          ; Команда NOP (No Operation) – немає
                     ; операції, рядок помічено міткою M1
M2:     NOP          ; Операції немає, рядок помічено міткою M2
M3:     JMP M4       ; Команда безумовного переходу на мітку M4
        MOV AX,1234H ; Завантаження до регістра AX числа 1234H
        MOV BX,5678H ; Завантаження до регістра BX числа 5678H
        CMP AX,BX    ; Порівняння вмісту регістра AX з вмістом
                     ; регістра BX
        JZ M1        ; Умовний перехід на мітку M1, якщо результат
                     ; порівняння дорівнює 0 (операнди рівні).
                     ; Унашому випадку умовний перехід не
                     ; відбувається, тому що операнди не рівні і
                     ; прапорець ZF скинуто у 0
        JNS M2       ; Умовний  перехід на мітку M1, якщо результат
                     ; порівняння додатний 0 (прапорець SF = 0). У
                     ; нашому випадку не відбувається, тому що
                     ;          _1234H
                     ;           5678H
                     ;           BBBCH    (прапорець SF = 1)
        JLE M3       ; Умовний перехід на мітку M3, якщо вміст
                     ; операнда dst (приймач) менший за операнд
                     ; src (джерело). В нашій програмі відбувається.
```

Команди виклику і повернення з підпрограми

CALL (CALL) – команда виклику підпрограми (процедури). Може відповідати близькому або далекому переходу до підпрограми, в залежності від початкової адреси розміщення підпрограми. При зверненні до підпрограми в стеку зберігається адреса повернення до основної програми (адреса комірки пам’яті в якій розміщується команда CALL).

Команда має вигляд

CALL ADDR.

Адреса переходу до підпрограми може задаватися міткою, вмістом регістра або комірки пам’яті.

Мітка (вміст регістра або комірки пам’яті) може відповідати зміщенню в поточному сегменті команд, куди передається управління. При цьому змінюється лише вміст EIP/IP. Якщо здійснюється далекий перехід, то змінюється вміст EIP/IP і вміст регістра CS.

RET (RETurn) – команда повернення з підпрограми до основної програми.

Команда має вигляд

RET.

Команда відновлює вміст регістра EIP/IP, шляхом завантаження із стека, значення збереженого при виконанні команди CALL.

Команди виклику і повернення з підпрограми не змінюють значення регістру ознак.

9.3.5 Команди організації циклів

Вхідний контроль:

1 Які види циклічних програм Ви знаєте?

2 В чому полягає різниця між арифметичним та ітераційним циклами?

3 В якому регістрі організується програмний лічильник циклів?

4 Наведіть приклади організації циклів будь-якою мовою високого рівня.

5 Покажіть на прикладах, як задається кількість повторень циклів будь-якою мовою високого рівня.

До команд управління циклом відносяться:

– команди організації циклу з лічильником ECX/CX;

– команди організації циклу з лічильником ECX/CX з можливістю дострокового виходу з циклу за додатковою умовою.

До першої групи відносяться команди:

JCXZ (Jump if CX=Zero) – перехід, якщо CX дорівнює 0;

JECXZ (Jump if ECX=Zero) – перехід, якщо ECX дорівнює 0.

Операндом в усіх командах слугує зміщення (мітка), за допомогою якого визначається адреса переходу.

Команди перевіряють вміст відповідного лічильника і якщо його вміст дорівнює 0, відбувається перехід на вказане у команді зміщення (мітку), а якщо не дорівнює, то виконується наступна команда.

До команд другої групи відносяться команди:

LOOP (LOOP control by register CX) – управління циклом за вмістом регістра CX;

LOOPE/LOOPZ (LOOP control by register CX not equal 0 and ZF=1) – управління циклом за вмістом регістра CX з урахуванням значення прапорця ZF;

LOOPNE/LOOPNZ (LOOP control by register CX not equal 0 and ZF=0) – управління циклом за вмістом регістра CX з урахуванням значення прапорця ZF.

Команда LOOP <адреса> забезпечує умовний перехід для циклічного виконання ділянки програми. Кількість повторень циклу визначається вмістом регістра ECX/CX. Усі різновиди команди LOOP автоматично виконують декремент вмісту ECX/CX і зупиняють виконання циклу, якщо вміст лічильника дорівнює 0.

Команди LOOPE/LOOPZ є різновидами однієї команди, так само як і команди LOOPNE/LOOPNZ. Алгоритм виконання цих команд однаковий. Команди декрементують вміст ECX/CX й аналізують його вміст і значення прапорця ZF, якщо вміст ECX/CX дорівнює 0, то виконується наступна за LOOPxx команда, якщо вміст ECX/CX дорівнює 1, то виконується перехід до початку циклу. Якщо значення ZF = 0, то команди LOOPE/LOOPZ виконують вихід з циклу, а команди LOOPNE/LOOPNZ повертаються до початку циклу. Для значення ZF = 1 команди виконується навпаки. Команди LOOPNE/LOOPNZ можливо використовувати для пошуку першого нульового елементу у рядку даних, якщо безпосередньо перед цією командою виконати порівнювання елемента з 0, а команди LOOPE/LOOPZ для пошуку першого ненульового елементу.

Контрольні запитання:

1 Які групи команд управління циклом Ви знаєте?

2 Що слугує операндом у командах управління циклом?

3 Як виконується декремент вмісту регістра-лічильника CX у програмі при використанні команд JCXZ і LOOP?

4 Які прапорці перевіряє команда LOOPE?

Контрольні запитання підвищеної складності:

1 Для чого використовується мітка в командах передачі управління?

2 Яка команда буде виконуватися після виконання команди LOOPNZ M1

```
M1: MOV AX,DX
    …
    LOOPNZ M1
    NOP,
```

якщо до її виконання в регістрі CX було записано число 0001H?

3 Як можливо використовувати команди LOOPE/LOOPZ при обробленні рядків даних?

4 Скільки разів буде виконуватись команда LOOPE M1, якщо до початку циклу в регістр CX був записаний нуль?

5 Наведіть фрагмент програми, в якому вихід з циклу здійснювався б за умовою JPO M1.

9.4 Створення програм на мові Асемблер-86

9.4.1 Лінійні програми

Вхідний контроль:

1 Які ділянки програм називаються лінійними?

2 Команда з якою адресою буде виконуватись після команди

7000:0100 MOV AX,7000H;?

У сучасних додатках рідко використовується програмне забезпечення, цілком написане мовою Асемблера. Частіше мова Асемблера використовується разом з мовою високого рівня, наприклад, C/C^{++}. Частина програми, написана мовою Асемблера, зазвичай призначена для керування периферійними пристроями, оскільки рішення таких задач на мовах високого рівня є більш складним і менш ефективним. Крім того, програми на мові Асемблер виконуються значно швидше, ніж написані будь-якою мовою високого рівня.

Основним принципом отримання ефективних програм є ефективний розподіл та використання ресурсів процесора. Оптимізація програми реалізується за рахунок мінімізації пересилань, використання вказівників на дані замість самих даних при роботі зі складними структурами, розміщення структури даних в одній локальній області пам'яті, розподіл регістрів для локальної ділянки програми тощо. Так, перш за все розподіляються спеціалізовані регістри, а решта виділяється для зберігання найбільш часто використовуваних даних. Дані, які використовуються одноразово, слід зберігати у пам’яті. Акумулятор виділяється для зберігання оперативних результатів.

Потрібна міра оптимальності програми визначається умовами її експлуатації і залежить від багатьох факторів. Для рідко використовуваних програм високий рівень оптимізації не є обов’язковий. Простіше за все оптимізація програми реалізується на лінійних послідовностях команд. У середині лінійних ділянок робиться присвоювання, виконуються операції над даними, вилучаються зайві операції за рахунок одноразового програмування загальних виразів з подальшим включенням їх у циклічні програми. Більш складною задачею є виконання еквівалентних адекватних перетворювань на лінійних ділянках з метою спрощення аналітичних виразів при збереженні достатньої точності та надійності обчислень.

Приклад 9.4.1 Ввести з портів з адресами 3F8H та 2F8H дані та запам’ятати їх у суміжних комірках пам’яті, починаючи з адреси 0020H у

сегменті даних. Округлити їх до 4-х розрядів, упакувати в один байт і зберегти у пам'яті за адресою 0030H у сегменті даних.

Під упакуванням розуміють розміщення у старшому півбайті першого числа, а в молодшому півбайті – другого числа.

Задача вирішується за виконанням наступного фрагмента програми:

```
MOV DX, 3F8H   ; Завантаження у регістр DX адреси порту 3F8H
MOV SI, 0020H  ; Завантаження в індексний регістр SI адреси комірки
               ; пам'яті для першого даного
MOV DI, 0030H  ; Завантаження у індексний регістр DI адреси комірки
               ; пам'яті для упакованого даного
MOV CL, 04H    ; Завантаження у CL кількості операції зсувів
XOR CH, CH     ; Обнулення регістра CH
IN AL, DX      ; Введення в акумулятор першого даного з порту
MOV CH, AL     ; Запам'ятовування першого даного у регістрі CH
MOV [SI], AL   ; Запам'ятовування першого даного у пам'яті
MOV DX, 2F8H   ; Завантаження у регістр DX адреси порту 2F8H
IN AL, DX      ; Введення в акумулятор другого даного з порту
INC SI         ; Нарощування адреси комірки пам'яті для другого
               ; даного
MOV [SI], AL   ; Запам'ятовування другого даного у пам'яті
AND CH, F0H    ; Округлення першого даного
AND AL, F0H    ; Округлення другого даного
SHR AL, CL     ; Переміщення округленого другого даного на місце
               ; другого півбайта
OR CH, AL      ; Упаковування даних в один байт
MOV [DI], CH   ; Запам'ятовування упакованих даних
NOP
```

Припустимо, що перше дане D1 становить 12H (00010010B), а друге D2 – 34H (00110100B). Округлення першого даного дасть півбайт, який дорівнює 0001B, а другого – 0011B. Упаковане дане складатиме 00010011B або 13H.

Програма складена з урахуванням подальшої можливості введення із портів масивів даних, їх упакування та запам'ятовування.

Приклад 9.4.2 Число -28H записати у регістри DL та DH у прямому коді двома способами. Наступні фрагменти програми виконують ці записи:

```
а) MOV DL, -28H        б) MOV DH, -28H
   NEG DL                 NOT DH
   OR DL,80H              ADD DH,81H
```

Запис будь-якого від'ємного числа здійснюється у регістр у доповняльному коді, тобто у регістрах DL та DH після виконання перших двох команд буде записане число -28H у доповняльному коді, тобто число D8H. Для

подання цього даного у прямому коді можна використати команду NEG DL, яка подасть це число без знака, а саме 28H, та команду OR DL,80H, яка установить 1 у старшому розряді. В результаті отримаємо прямий код числа -28H, який дорівнює A8H (фрагмент а)). У фрагменті б) число D8H, записане у регістр DH у доповняльному коді, інвертується і до нього додається число 81H. Ця операція установлює 1 у старшому розряді результату і додає 1 до молодшого розряду; у регістрі DL буде записане також число A8H.

Приклад 9.4.3 Охарактеризувати ознаками ZF, SF та PF дане, яке зберігається у регістрі AL. Команда

OR AL, AL

не змінює дане, яке зберігається в AL, і виставляє зазначені ознаки. У припущенні, що дане є число 56H, ознаки дорівнюватимуть:

ZF=0;
SF=0;
PF=1.

Приклад 9.4.4 Знайти вміст акумулятора та ознак CF, SF, AF після виконання фрагментів програм:

а)	б)	в)
MOV AL, 4EH	MOV AL, 2AH	MOV AL, 2AH
SUB AL, 2AH	SUB AL, 4EH	SUB AL, E4H
NOP	NOP	NOP

При виконанні віднімання отримуємо вміст акумулятора:

а)	б)	в)
_4EH	_2AH	_2AH
2AH	4EH	E4H
24H	DCH	46H
CF=0	CF=1	CF=1
AF=0	AF=1	AF=0
SF=0	SF=1	SF=0

Ознака CF установлюється в 1, якщо мала місце позика, тобто у фрагментах б) та в). При відніманні із меншого числа більшого ознака CF завжди установлюється в 1, а ознака знаку SF=1 тільки у разі отримання “коректного” результату, тобто за відсутності переповнення розрядної сітки. Для виявлення меншого з двох чисел слід користуватися ознакою CF, якщо аналізується результат виконання команд віднімання або порівняння.

Приклад 9.4.5 При завантаженні комп'ютера тестуються наявні асинхронні адаптери й ініціалізуються перші два з них: COM1 та COM2, які мають базові адреси 0000:0400H та 0000:0300H. Наступний фрагмент програми дозволяє прочитати з регістра керування з адресою 3FBH порту COM1 байт стану режиму адаптера:

```
MOV DX, 3FBH ; Завантаження адреси керувального регістра у DX
IN AL, DX    ; Введення поточного режиму адаптера
```

При правильній ініціалізації байт стану дорівнюватиме 000X0011B = 03H або 13H. Це відповідає такому режиму: немає контролю на парність (D3 = 0, D4 = X), один стоповий біт (D2 = 0), кількість бітів дорівнює 8 (D1 = 1, D2 = 1).

Приклад 9.4.6 Два байти D1 та D2 є 3- та 5-м елементами масиву, що починається з ефективної адреси 0010H. Інвертувати перший байт, вирізати 0, 2, 7 розряди другого; отримані результати зберегти у стеку. Припустимо, що перший байт D1 дорівнює 25H, а другий D2 – 73H. Виконаємо вказані операції порозрядно та вкажемо ознаки результату (прапорці) на кожному кроці виконання. Сегменти стека та даних є суміщені.

$F1 = (\overline{D1}) = \overline{00100101} = 11011010 = DAH.$

F2 = D2 /\ mask; mask = 01111010 = 7AH.

$$F2 = /\backslash \begin{array}{r} 01110011 \\ \underline{01111010} \\ 01110010 \end{array} \qquad F2 = 72H \quad OF = 0, CF = 0, SF = 0, \; AF = 0, ZF = 0, PF = 1.$$

Задамо довільно 16 елементів масиву байтів, з яких третій дорівнює 25H, а п'ятий дорівнює 73H.

7000:0010 12 34 56 25 43 73 43 54 65 76 82 37 15 13 14 61 52.

Вказівник стека задамо таким, який дорівнює 2080H.
Структурна схема програми наведена на рис. 9.12.
Фрагмент програми, яка вирішує цю задачу, має вид

```
MOV AX, 7000H      ; Завантаження
MOV DS, AX         ; сегментних
MOV SS, AX         ; регістрів
MOV SP, 2080H      ; Завантаження вказівника стека
MOV BX, 0010H      ; Завантаження базового регістра
MOV AL, [BX+03]    ; Пересилання першого байта до AL
NOT AL             ; Інвертування першого байта
```

```
MOV AH, [BX+05]     ; Пересилання другого байта до AH
AND AH, 7AH         ; Змінення другого байта
PUSH AX             ; Запам'ятовування результатів у стеку
```

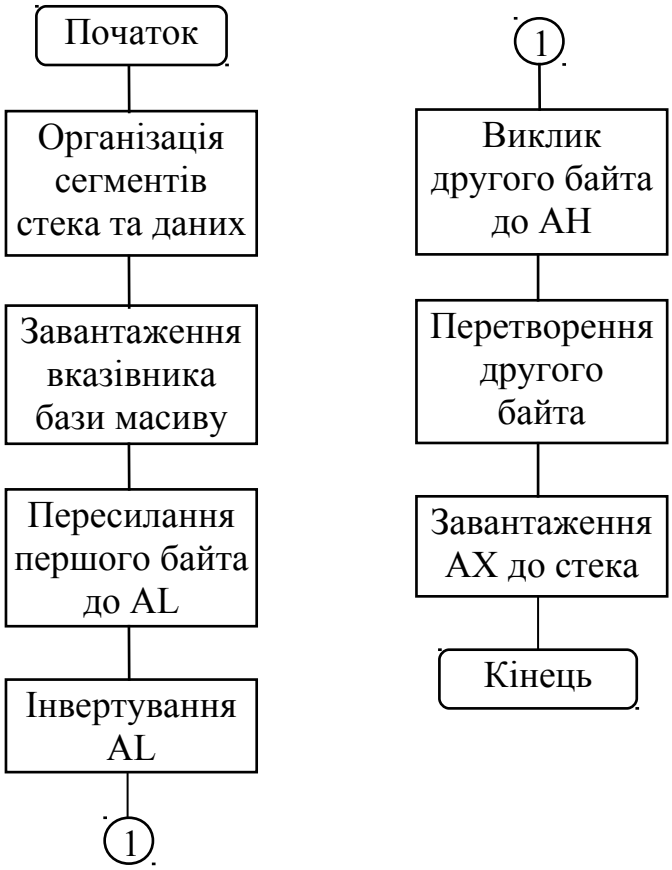


Рисунок 9.12 – Структурна схема програми

Програма виконується так.

Усі команди пересилань та команда NOT не змінюють прапорці; результатом виконання команди NOT є число DAH, що вміщується у регістрі AL, а результатом команди AND є число 72H, що вміщується в AH; ця команда установлює прапорці OF = 0, CF = 0, SF = 0, AF = 0, ZF = 0, PF = 1.

Приклад 9.4.7 Виконати операцію виключного АБО над двома словами, D1 та D2, що розміщені у пам'яті. Помножити результат на беззнакове число 33H і розташувати добуток на місці другого слова. Перше знаходиться у масиві, що починається зі зміщення 10H та адресоване вмістом індексного регістра SI, а друге знаходиться у масиві, що починається зі зміщення 20H та адресується вмістом індексного регістра DI і зміщенням 2H. Початкова адреса сегмента даних є 7000H.

Створимо два масиви, починаючи зі зміщення 10H та 20H відповідно. Припустимо, що у регістрі SI міститься число 6H, а у регістрі DI – число 2H.

```
7000:0010   12 34 56 25 40 43 71 43 65 76 82 37 15 13 18 60
7000:0020   14 36 58 27 45 73 48 54 65 76 82 39 17 15 14 61
```

Тоді перше слово D1 становить 4371H, а друге D2 дорівнює 7345H. Перший проміжний результат F1 становить:

F1=D1⊕D2=3034H.

$$F1 = \oplus \begin{array}{l} 0100001101110001 \\ 0111001101000101 \\ \hline 0011000000110100 \end{array}$$

Ознаки результату становлять OF = 0, CF = 0, SF = 0, AF = 0, ZF = 0, PF = 0.

Другий результат F2 становить:

F2 = F1 × 33H = 3034H × 33H = 99A5CH.

Ознаки результату дорівнюють: OF = 1, CF = 1, SF = 1, AF = 1, ZF = 0.

Програма має вигляд:

```
MOV AX, 7000H          ; Організація
MOV DS, AX             ; сегмента даних
MOV BX, 0010H          ; Завантаження до BX початкової адреси першого
                       ; масиву
MOV DI, 2H             ; Завантаження
MOV SI, 6H             ; індексних регістрів
XOR DX, DX             ; Обнулення DX
MOV AX, [BX+SI]        ; Завантаження першого слова до AX
MOV BX, 0020H          ; Завантаження до BX початкової адреси другого
                       ; масиву
XOR AX, [BX+DI+2]      ; Складання за модулем 2 з другим словом
MOV CX, 33H            ; Завантаження множника до CL
MUL CX                 ; Множення на константу 33H
MOV [BX+DI+2], AX      ; Пересилання молодшого слова добутку на місце
                       ; другого слова
MOV [BX+DI+4], DX      ; Пересилання старшого слова добутку до другого
                       ; масиву
```

Після виконання програми в регістр AX буде записане число 9A5CH, а в регістр DX число 0009H.

Контрольні запитання:

1 Як подаються числа беззнакові та зі знаком у МП сім'ї Intel?

2 Подані в яких системах числення дані можуть оброблюватись у МП сім'ї Intel?

3 Четвертий елемент першого масиву дорівнює 32H. Поділити його на п'ятий елемент другого масиву, що дорівнює 4H, усіма відомими вам способами.

4 Інвертувати слово, що становить п'ятий елемент першого масиву; поділити результат на третій елемент другого масиву. Частку разом з регістром прапорців запам'ятати у стеку.

5 Знайти логічний добуток четвертого байта першого масиву та третього байта другого масиву. Результат помножити на 44H, добуток запам'ятати за ефективною адресою, що визначається вмістом регістра DI.

6 Заповнити масив з чотирьох слів, починаючи з адреси DS:50, використовуючи команду STOSW.

7 Порівняти нульовий елемент першого масиву з другим елементом другого масиву. Запам'ятати у стеку вміст регістра прапорців та суму вказаних елементів.

Контрольні запитання підвищеної складності:

1 Завантажити одночасно регістр сегмента даних та акумулятор двобайтовими елементами першого масиву, починаючи з другого елемента за допомогою прямого адресування.

2 Сумістити сегмент даних та додатковий сегмент даних.

9.4.2 Розгалужені програми

Вхідний контроль:

1 За якими умовами (прапорцями) можна реалізувати умовні переходи?

2 Напишіть команду безумовного переходу на адресу команди, яка знаходиться в іншому сегменті кодів.

У послідовних ділянках програми виконування команд відбувається згідно з їхнім розташуванням. Однак за умовами тієї чи іншої задачі може виникнути необхідність зміни порядку виконання команд. Це реалізується за допомогою команд, призначених для передачі керування з однієї точки програми до іншої. При виконанні будь-якої команди відбувається нарощування вмісту вказівника команд IP, що дозволяє одержати адресу наступної виконуваної команди у лінійних програмах. Якщо ж треба перейти до потрібної адреси у програмі, використовують безпосередню зміну значення вказівника команди IP та/або значення сегментного регістра коду CS за допомогою команд умовного та безумовного переходів. Розрізняють команди дальнього (far), близького (near) та короткого (short) переходів. При дальньому переході змінюється не тільки вміст вказівника команд IP, але й вміст сегментного регістра коду CS, тобто можливий перехід до будь-якої комірки

пам’яті, на відстань, більшу ніж 2^{16} адрес комірок пам’яті, а не тільки у межах сегмента коду. Це дає можливість реалізувати міжсегментні переходи до будь-якого сегмента, наприклад, при виконанні підпрограм. Цей сегмент потім стає поточним сегментом коду. При близькому переході змінюється тільки вміст вказівника команд IP у межах [-2^{16}... $+2^{16}-1$] адрес, а вміст сегментного регістра CS не змінюється, тобто перехід можливий тільки у сегменті коду CS. При короткому переході межі змінення вказівника команд IP становлять [-128...+127] адрес.

Безумовний перехід

Команди безумовного переходу (БП) дозволяють перейти на задану адресу програми без запам’ятовування адреси повернення. Є можливими три форми мнемонічного подання команди безумовного переходу:

```
JMP 0110H          ; Прямий перехід на адресу 0110H
JMP [offset]       ; Перехід за зміщенням відносно поточного вмісту
                   ; вказівника команд IP
JMP [operand]      ; Непрямий перехід; адреса переходу вміщується до
                   ; регістрів загального призначення або до комірок
                   ; пам’яті
```

При прямому переході адреса переходу замінює вміст вказівника команд і, за необхідності, вміст сегментного регістра коду.

Зміщення являє собою 8-, 16- та 32-розрядне число (останнє тільки для МП I80386 та старших).

Зміщення вважається знаковим, тому є можливий перехід у бік початку або кінця програми. Змінюється тільки вказівник команд IP, а сегментний регістр CS залишається незмінним.

При непрямому переході адреса продовження програми визначається вмістом одного із 16- (AX, BX, CX, DX, SI, DI, BP, SP) або 32- (EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP – для старших моделей) розрядних регістрів загального призначення або ділянки пам’яті. У першому випадку реалізується короткий або близький перехід, а у другому і третьому випадках перехід може бути також і далеким. Вміст вказаного у команді регістра або ділянки пам’яті є новим вмістом вказівника команд IP та за необхідності сегментного регістра коду CS. У наступних фрагментах програми показані можливі варіанти указання адреси при безумовних переходах.

```
  CS : IP
7000 : 30 MOV DS, AX        ; Завантаження DS  з AX
7000 : 32 MOV AX,1234H      ; Завантаження AX даним 1234H
7000 : 35 JMP 0039 H        ; Обхід команди MOV CX,AX
7000 : 37 MOV CX,AX         ; Завантаження CX з AX
7000 : 39 MOV DX,AX         ; Завантаження DX з AX
```

```
  CS : IP
7000 : 30 MOV DS, AX          ; Фрагмент
7000 : 32 MOV AX, 0039H       ; програми
7000 : 35 JMP AX              ; аналогічний
7000 : 37 MOV CX, AX          ; попередньому
7000 : 39 MOV DX, AX

  CS : IP
7000 :  30 MOV AX, 003CH      ; Фрагмент
7000 :  33 MOV [ 50], AX      ; програми
7000 :  36 JMP [50]           ; аналогічний
7000 :  3A MOV CX, AX         ; попередньому
7000 :  3C MOV DX, AX
```

Прапорці при виконанні команд безумовних переходів не змінюються.

Умовні переходи

Умовні переходи (УП) у програмах здійснюються при виконанні вказаних у них умов залежно від одержаного перед УП результату і не реалізуються, якщо зазначені умови не виконані, тоді виконується ділянка програми, що йде безпосередньо за командою умовного переходу. Команди умовних переходів наводяться у табл. 9.4. Для деяких умов існують кілька команд умовного переходу. Використання певної команди залежить тільки від програміста. Умови “вище” або “нижче” відносяться до величин без знака, а “більше” та “менше” до величин зі знаком. Зміщення у командах умовного переходу – це величини у діапазоні -128...+127, тобто вони займають 1 байт. Для МП I80386 та старших можна використовувати зміщення, що займає 2 або 4 байти. Вміст регістра CS не змінюється.

Якщо ж потрібна адреса знаходиться поза даним сегментом коду, використовують команду умовного переходу з протилежною умовою, а за нею вміщують вже команду безумовного переходу з необхідною адресою.

Достатньо розповсюдженим є випадок, коли розгалуження програм виконується залежно від результату порівняння двох операндів за допомогою команди CMP – порівняння як знакових, так і беззнакових чисел.

Приклад 9.4.8 Два однобайтових беззнакових даних D1 та D2 зберігаються у стеку. Перше дане міститься за молодшою адресою стека, друге – за старшою. Якщо різниця між першим та другим даними є негативна і кількість одиниць в ній непарна, то слід:

– збільшити перше дане на 5H;

– інвертувати друге дане;

– отримані результати запам’ятати відповідно за суміжними адресами у сегменті даних зі зміщеннями 10H та 11H.

У разі парної кількості одиниць у різниці даних необхідно:
– збільшити перше дане на 02H;
– збільшити друге слово на 01H;
– отримані результати записати за тими самими адресами.
Якщо різниця даних позитивна, їх слід зберегти у стеку.
Структурна схема алгоритму має вид (рис. 9.13).

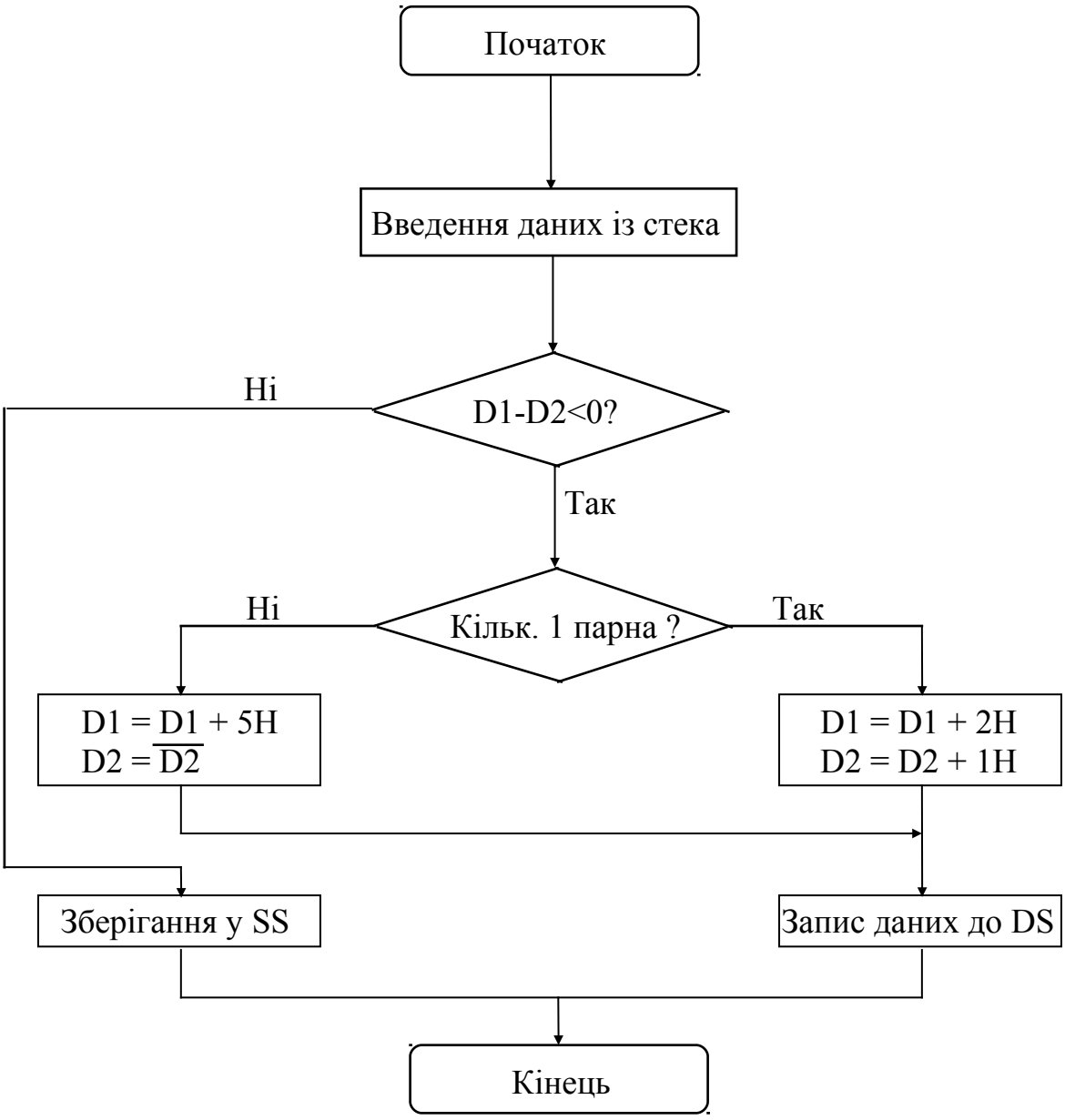


Рисунок 9.13 – Структурна схема алгоритму

Фрагмент програми розв’язання задачі:

```
MOV AX, 0F0A      ; Завантаження даних до AX
PUSH AX           ; Завантаження даних до стека
POP AX            ; Завантаження даних із стека до AX
MOV BX,AX         ; Збереження вмісту AX в BX
SUB BL,BH         ; Різниця даних
```

```
        JNS POSIT       ; позитивна ?
        JP EVEN         ; Ні, кількість одиниць парна?
        ADD AL, 05H     ; Ні, зміна першого слова
        NOT AH          ; Зміна другого даного
        MOV [10H], AX   ; Запам'ятовування нових даних у
                        ; сегменті даних
        JMP KINEC       ; БП на кінець програми
EVEN:   ADD AL, 02H     ; Так, зміна першого даного
        INC AH          ; Інкрементування другого даного
        MOV [10], AX    ; Запам'ятовування даних у сегменті даних
        JMP KINEC       ; БП на кінець програми
POSIT:  PUSH AX         ; Запам'ятовування вхідних даних у стеку
KINEC:  NOP             ; Кінець програми
```

Перед виконанням програми до стека, враховуючи значення вказівника стека SP, треба по черзі завантажувати дані D1 та D2 так, щоб кожного разу виконувалася одна гілка розгалуженої програми , а комірки пам'яті у сегменті даних зі зміщеннями 10H та 11H попередньо обнулити, щоб перевірити правильність виконання програми. Якщо перше дане дорівнює 0AH, а друге – 0FH , то при завантаженні регістра AX із стека після виконання команди POP AX у регістрі AL буде розміщене перше дане – 0AH, а у регістрі AH – друге дане, 0FH. Різниця першого та другого даних після виконання команди SUB BL, BH становить FBH. Установлюється ознака SF = 1, що вказує на одержання негативного результату. Через це буде виконуватися команда JP, чергова за списком, а умовний перехід за командою JNS не виконується.

Кількість одиниць у різниці непарна, тому установлюється ознака (прапорець) PF = 0, і виконується лінійна ділянка програми, що йде за командою УП JP, включаючи команду БП на кінець програми. Сам умовний перехід не виконується. У результаті виконання цього фрагмента у комірці пам'яті зі зміщенням 10H буде записане число 00H, а у комірці пам'яті зі зміщенням 11H – число F0H.

Якщо вхідні дані становили відповідно 0FH та 0AH, то їхня різниця буде позитивною – 05H, і після її обчислення здійснюється перехід до команди з ефективною адресою 012F PUSH AX, за якою вхідні дані будуть розміщені у стеку за попередніми адресами.

Якщо вхідні дані дорівнюють відповідно 0AH та 0EH, то в їх різниці FCH буде парна кількість одиниць, і за ознакою PF = 1 команда JP 0126 реалізує умовний перехід. У результаті виконання послідовної ділянки програми, включаючи команду БП на кінець програми, у комірці пам'яті зі зміщенням 10H буде завантажене число 0CH, а зі зміщенням 11H – число 0FH.

У табл. 9.5 ... 9.7 подані протоколи виконання різних віток розгалуженої програми залежно від різних значень вхідних даних D1 та D2. Для наочності у стовпчику IP показано адресу поточно виконуваної команди, а не наступної.

Слід зазначити, що приклад має виключно навчальне значення.

Таблиця 9.5 – Протокол виконання програми D1 = 0AH, D2 = 0FH

№ пп.	Мнемоніка команд	IP	AX	BX	SP	DS	SS	CS	PF	SF	ZF	CF
1	MOV AX,0F0A	0109	OF0A	0000	FFEE	7000	7000	7000	1	0	0	0
2	PUSH AX	010C	0F0A	0000	FFEC	7000	7000	7000	1	0	0	0
3	POP AX	010D	0F0A	0000	FFEE	7000	7000	7000	1	0	0	0
4	MOV BX, AX	010E	0F0A	0F0A	FFEE	7000	7000	7000	1	0	0	0
5	SUB BL, BH	0110	0F0A	0FFB	FFEE	7000	7000	7000	0	1	0	1
6	JNS 0128	0112	0F0A	0FFB	FFEE	7000	7000	7000	0	1	0	1
7	JP 011F	0114	0F0A	0FFB	FFEE	7000	7000	7000	0	1	0	1
8	ADD AL, 05	0116	0F0F	0FFB	FFEE	7000	7000	7000	1	0	0	0
9	NOT AH	0118	F00F	0FFB	FFEE	7000	7000	7000	1	0	0	0
10	MOV[10], AX	011A	F00F	0FFB	FFEE	7000	7000	7000	1	0	0	0
11	JMP 0129	0126	F00F	0FFB	FFEE	7000	7000	7000	1	0	0	0
12	NOP	0129	F00F	0FFB	FFEE	7000	7000	7000	1	0	0	0

Таблиця 9.6 – Протокол виконання програми D1 = 0FH, D2 = 0AH

№ пп.	Мнемоніка команд	IP	AX	BX	SP	DS	SS	CS	PF	SF	ZF	CF
1	MOV AX,0A0F	0109	0A0F	0000	FFEE	7000	7000	7000	0	0	0	0
2	PUSH AX	010C	0A0F	0000	FFEC	7000	7000	7000	0	0	0	0
3	POP AX	010D	0A0F	0000	FFEE	7000	7000	7000	0	0	0	0
4	MOV BX, AX	010E	0A0F	0A0F	FFEE	7000	7000	7000	0	0	0	0
5	SUB BL, BH	0110	0A0F	0A05	FFEE	7000	7000	7000	1	0	0	0
6	JNS 0129	0112	0A0F	0A05	FFEE	7000	7000	7000	1	0	0	0
7	PUSH AX	0128	0A0F	0A05	FFEC	7000	7000	7000	1	0	0	0
8	NOP	0129	0A0F	0A05	FFEC	7000	7000	7000	1	0	0	0

Таблиця 9.7 – Протокол виконання програми D1 = 0AH, D2 = 0EH

№ пп.	Мнемоніка команд	IP	AX	BX	SP	DS	SS	CS	PF	SF	ZF	CF
1	MOV AX,0E0A	0109	0E0A	0000	FFEE	7000	7000	7000	0	0	0	0
2	PUSH AX	010C	0E0A	0000	FFEC	7000	7000	7000	0	0	0	0
3	POP AX	010D	0E0A	0000	FFEE	7000	7000	7000	0	0	0	0
4	MOV BX, AX	010E	0E0A	0E0A	FFEE	7000	7000	7000	0	0	0	0
5	SUB BL, BH	0110	0E0A	0EFC	FFEE	7000	7000	7000	1	1	0	1
6	JNS 0128	0112	0E0A	0EFC	FFEE	7000	7000	7000	1	1	0	1
7	JP 011F	0114	0E0A	0EFC	FFEE	7000	7000	7000	1	1	0	1
8	ADD AL,02	011F	0E0C	0EFC	FFEE	7000	7000	7000	0	0	0	0
9	INC AH	0121	0F0C	0EFC	FFEE	7000	7000	7000	1	0	0	0
10	MOV[0010],AX	0123	0F0C	0EFC	FFEE	7000	7000	7000	1	0	0	0
11	JMP 0129	0126	0F0C	0EFC	FFEE	7000	7000	7000	1	0	0	0
12	NOP	0129	0F0C	0EFC	FFEE	7000	7000	7000	1	0	0	0

Контрольні запитання:

1 З якою метою у програмах реалізуються БП?

2 Чи зберігається адреса повернення до основної програми, якщо реалізується безумовний перехід?

3 Знайти добуток двох однобайтових даних, що зберігаються у регістрі AX. Якщо одержаний результат від'ємний і вміщує парну кількість одиниць, то до нього треба додати одиницю і запам'ятати у стеку. Якщо кількість одиниць непарна, то добуток треба подати у прямому коді і записати у сегменті даних за зміщенням 0012. Якщо добуток додатний, його треба запам'ятати у сегменті даних за адресою 7000:0014H. Значення вхідних даних задати самостійно.

Контрольні запитання підвищеної складності:

1 Знайти частку від ділення двох однобайтових даних. Ділене знаходиться у регістрі AH, а дільник у регістрі AL. Якщо одержаний результат від'ємний і вміщує парну кількість одиниць, то до нього треба додати одиницю і запам'ятати у стеку. Якщо кількість одиниць непарна, то частку треба подати у прямому коді за зміщенням 0012. Якщо частка додатна, її треба запам'ятати у сегменті даних за адресою 7000:0016H. Значення вхідних даних задати самостійно.

2 Перевірити, чи буде вірний фрагмент програми

```
7000:0120       CMP AL,BL
7000:0122       JZ 8000:0100
```

9.4.3 Циклічні програми

Вхідний контроль:

1 Чи завжди у циклічній частині програми треба вказувати необхідну кількість повторень циклу?

2 Наведіть приклади організації циклів за допомогою команд мови Асемблер і покажіть, як вони реалізуються.

Програми, у яких багаторазово виконуються ті ж самі ділянки обчислень, як правило, з різними значеннями вхідних даних, називаються **циклічними**, а послідовність команд, що повторюються, **циклами**. Цикли з наперед заданою кількістю повторень називаються **арифметичними**. Цикли, що не мають заздалегідь заданої кількості повторень, називаються **ітераційними**. Вихід з них відбувається при виконанні заданої умови. Для реалізації арифметичних циклів треба організувати лічильник циклів. За умовчанням для організації лічильника циклів призначений регістр CX (ECX у МП I80386 та старших). Якщо лічильник спадний, то при виконанні кожного циклу із його вмісту віднімається одиниця. Цикли повторюються доти, доки вміст лічильника буде більший за нуль. Циклічна програма вважається завершеною, якщо лічильник дорівнює нулю. Рідше використовуються зростаючі лічильники. При роботі з масивами закінчувати цикли можна також при досягненні поточної адреси

елемента масиву адреси його останнього елемента. Як лічильник циклів можна використовувати будь-який регістр загального призначення, але у системі команд мови Асемблера існують спеціальні команди умовного переходу, в яких перевіряється вміст регістра CX (ECX).

Приклад 9.4.9 Перед записом байта даних у регістр передавача послідовного асинхронного адаптера RS-232-C треба перевірити, чи вільний є регістр зберігання передавача, тобто чи завершено передавання попереднього символу. Ознакою “порожнього” регістра зберігання є установлений в 1 п’ятий біт регістра стану лінії з адресою 3FDH. Наступний фрагмент програми реалізує цю перевірку у циклі:

```
     MOV DX, 3FDH; Завантаження адреси регістра стану лінії
M1:  IN AL, DX     ; Введення байта стану лінії
     AND AL, 20H   ; Регістр зберігання передавача
     JZ M1         ; порожній, D5 = 1?
     NOP
```

Приклад 9.4.10 Перед прийомом даних з порту приймача послідовного асинхронного адаптера RS-232-C треба перевірити, чи прийняте з лінії дане перебуває у буферному регістрі приймача. Ознакою цього є наявність 1 у нульовому біті регістра стану лінії з адресою 3FDH. Наступний фрагмент програми реалізує цю перевірку в циклі:

```
     MOV DX, 3FDH; Завантаження адреси порту стану лінії
M1:  IN AL, DX     ; Введення байта стану лінії
     AND AL, 01H   ; Перевірка наявності даного у буферному регістрі
                   ; приймача
     JZ M1         ; D0 = 1
```

Приклад 9.4.11 Знайти у чотириелементному рядку байт, що дорівнює 43H, якщо рядок розміщений, починаючи з ефективної адреси 0200H, а вміст сегментного регістра даних ES дорівнює 7000H:

7000:0200 41 42 43 44.

Лічильник байтів у рядку організуємо у регістрі CL, ефективна адреса байтів вміщується в індексний регістр DI, а еталон шуканого байта завантажуємо до AL. Для перепускання неспівпадаючих елементів використовується префікс REPNZ.

```
MOV AX, 7000H   ;
MOV ES, AX      ;
MOV CL, 0004H   ; Організація лічильника циклів
MOV DI, 0200H   ; Організація непрямого регістрового адресування
```

```
                    ; елементів рядка
MOV AX, 0043H       ; Завантаження еталонного байта до AL
STD                 ; Установлення прапорця напрямку
CLD                 ; Скидання прапорця напрямку
REPNZ               ; Організація перепускання неспівпадаючих елементів
SCASB               ; Сканування байтів рядка
```

Програма виконується циклічно чотири рази, кожний цикл являє собою лінійну програму. Після сканування усіх байтів рядка в регістрі CX буде записане число 0001H, а в регістрі DI – число 0203H, на одиницю більше, ніж адреса шуканого байта. Прапорець ZF установлюється після виконання програми.

Приклад 9.4.12 Одновимірний масив M(N), N = 10H однобайтових елементів зі знаками розміщений у сегменті даних, починаючи з ефективної адреси 20H; розсортувати елементи масиву на додатні M(P) та від'ємні M(NEG) і запам'ятати додатні елементи в області пам'яті MP, починаючи з ефективної адреси 30H, а від'ємні елементи в області пам'яті MNEG, починаючи з ефективної адреси 40H. Сегменти DS та ES суміщені.

Структурна схема алгоритму наведена на рис. 9.14.

Фрагмент програми розв'язання задачі.

```
        MOV CX, 0010H   ; Завантаження лічильника циклів
        MOV SI, 0020H   ; Завантаження адреси первісного масиву
        MOV DI, 0030H   ; Завантаження початкової адреси масиву
                        ; додатних елементів
        MOV BX, 0040H   ; Завантаження початкової адреси масиву
                        ; від'ємних елементів
        CLD             ; Виставлення автоінкрементування SI та DI
CYCLE:  LODSB           ; Виклик чергового елемента  масиву
        OR AL, AL       ; Виставлення ознак елемента масиву
        JS NEG          ; Елемент є додатний?
P:      STOSB           ; так, запис до масиву додатних чисел
        JMP COUNT       ; БП на команду перевірки лічильника
                        ; циклів
NEG:    MOV [BX], AL    ; ні, запис до масиву  чисел
        INC BX          ; Нарощування адреси масиву від'ємних
                        ; чисел
COUNT:  LOOP CYCLE      ; Перевірка лічильника циклів
        NOP
```

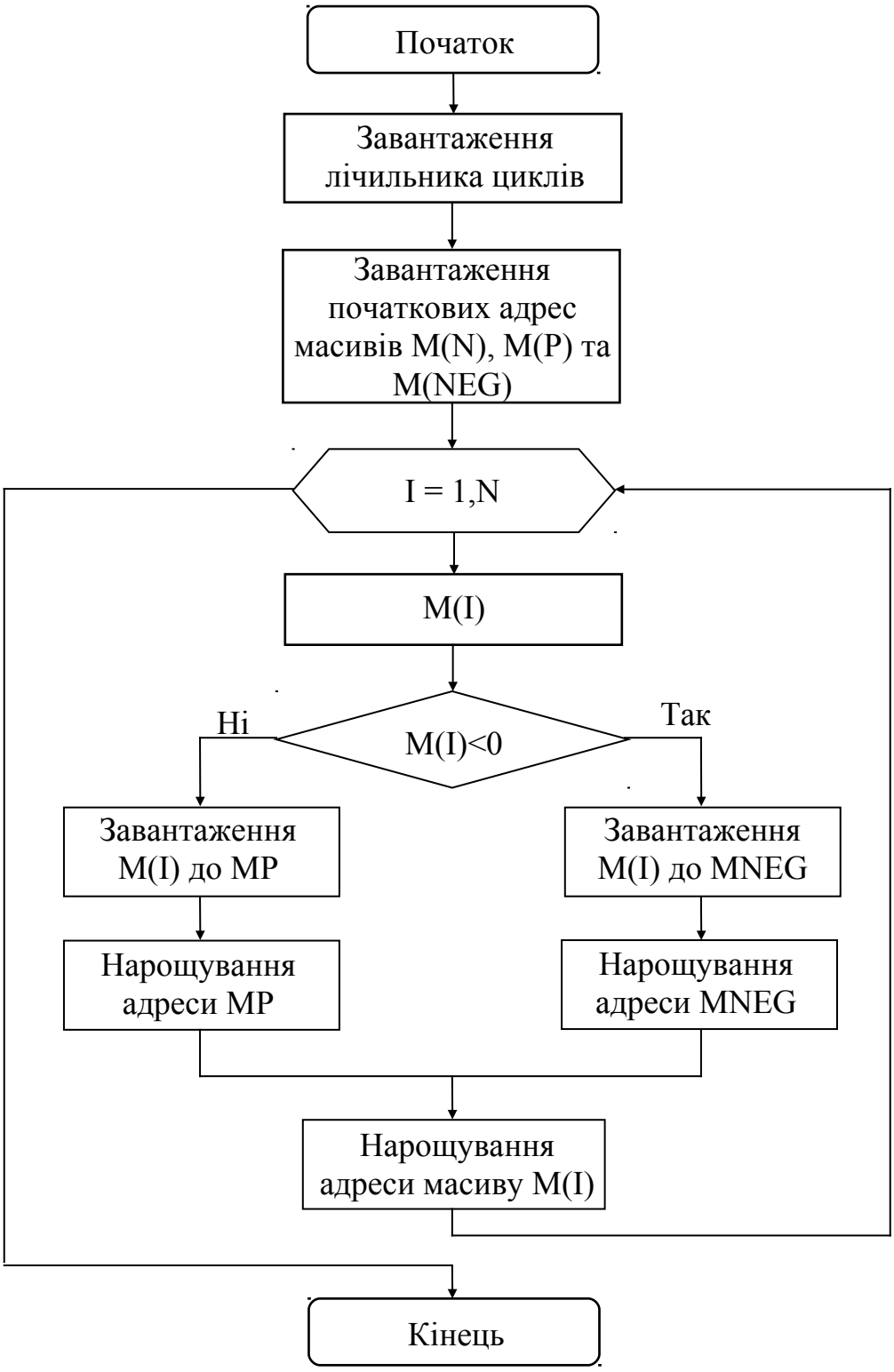


Рисунок 9.14 – Структурна схема алгоритму

Приклад 9.4.13 Знайти середнє арифметичне N двобайтових елементів масиву M(N), N = 10H, що вони зберігаються у пам’яті в доповняльному коді, починаючи з адреси 7000:0020. Результат розмістити за ефективною адресою 30H. Структурна схема алгоритму подана на рис. 9.15.

Рисунок 9.15 – Структурна схема програми

Програма розв’язання задачі.

```
MOV AX, 7000H     ; Організація сегмента даних,
MOV DS, AX        ; починаючи з ефективної адреси 7000:0000
```

```
        MOV BX, 0020H    ; Завантаження адреси початку масиву
        MOV AX, 0000H    ; Обнулення акумулятора (S = 0)
M1:     ADD AX, [BX]     ; Додавання чергового елемента масиву
        ADD BX, 2        ; Нарощування адреси чергового елемента
                         ; масиву
        CMP BX, 0030H    ; Чи дорівнює адреса чергового елемента масиву
        JNZ M1           ; адресі останього+2? Якщо ні, повернення на
                         ; початок циклу
        MOV BX, 0010H    ; Завантаження дільника до BX
        IDIV BX          ; Визначення середнього арифметичного
        MOV [30H], AX    ; Запам’ятовування середнього арифметичного
        NOP
```

Сума елементів масиву вміщується в акумуляторі AX, результат треба запам’ятовувати за ефективною адресою 30H.

Приклад 9.4.14 Написати підпрограму часової затримки тривалістю 100 мкс при умові, що тривалість такту 0,1 мкс. Для організації підпрограми затримки, яка найчастіше має назву DELAY, треба використати лічильник для зберігання числа повторень циклів, від якого залежить час затримки. Значення X обчислюється за формулою

$$X = \left] \frac{t_ç - t_0}{t_ö} \right[,$$

де дужки] [означають, що дрібна частина відкидається; $t_з$ – задане значення затримки; t_0 – час, потрібний для одноразово виконуваних команд; $t_ц$ – час, потрібний для виконання циклічно повторюваних команд. Вибір застосовуваних у підпрограмі команд та обчислення X являє собою ітераційний процес, тривалість якого залежить від необхідної точності затримки. За наявності високих вимог до точності забезпечуваної затримки можливі два шляхи розв’язання цієї задачі:

1) зменшують отримане значення X на кілька одиниць, а отримане зменшення компенсують командою NOP, яка виконується багаторазово;

2) змінюють значення $t_ц$ за рахунок включення у цикл інших команд.

Наступний фрагмент підпрограми DELAY дає уявлення про принцип програмної реалізації часової затримки:

```
        CALL DELAY   ; 19 тактів при умові внутрішньосегментного
                     ; переходу
DELAY:  MOV AL, X    ; 2 такти, X обчислюється за формулою та
                     ; задається у команді з безпосереднім
                     ; адресуванням
TIME:   DEC AL       ; 3 такти
```

```
        JNZ TIME        ; 8 тактів
        NOP             ; 2 такти
        RET             ; 8 тактів
```

На початку підпрограми DELAY, як і в кожній підпрограмі, треба запам'ятати вміст використовуваних регістрів та регістра прапорців:

```
        CALL DELAY      ; 19 тактів
DELAY:  PUSH BX         ; 11тактів
        PUSHF           ; 10 тактів
        MOV BL, X       ; 2 такти, X повинен завантажуватись у BL
                        ; як операнд з безпосереднім адресуванням
M1:     DEC BL          ; 3 такти
        JNZ M1          ; 16 тактів
        POP F           ; 8 тактів
        POP BX          ; 8 тактів
        RET             ; 8 тактів
```

$t_0 = (19+11+10+2+8+8+8)\cdot 0{,}1 = 7{,}3$ мкс;
$t_ц = (3+16)\cdot 0{,}1 = 1{,}9$ мкс;
$t_з = 100$ мкс.

$$X = \left\lfloor \frac{100 - 7{,}3}{1{,}9} \right\rfloor = 49D = 31H$$

При X = 49D отримаємо $t_з = 6{,}6 + 49 * 1{,}9 = 6{,}6 + 93{,}1 = 99{,}7$ мкс.

Така точність формування затримки не є достатня і тому до одноразово виконуваних команд треба додати команду, яка б займала 3 такти і виконувалась 0,3 мкс. Це може бути команда OR BL, BL. Остаточно фрагмент підпрограми затримки має вигляд:

```
DELAY: PUSH BX
       PUSHF
       MOV BL, 31H
M1:    DEC BL
       JNZ M1
       OR BL, BL
       POP F
       POP BX
       RET
```

Слід зазначити, що доцільно перед зверненням до підпрограми DELAY заборонити масковані переривання від зовнішніх пристроїв і наступною після команди CALL DELAY вжити команду дозволу переривань.

Контрольні запитання:

1 Які програми називаються циклічними?

2 Які види циклічних програм ви знаєте?

3 Які команди умовних переходів використовуються для пошуку в рядку першого нульового або ненульового елемента?

4 Який спосіб адресування використовується при роботі з двовимірними масивами?

5 У масиві N однобайтових чисел, розміщених у пам’яті, починаючи з адреси 7000: 0010H, підрахувати суму парних чисел. Результат розмістити за зміщенням 0027H.

6 У масиві M(N) однобайтових чисел, розміщених у пам’яті, починаючи з адреси 7000: 0010H, підрахувати суму непарних чисел. Результат розмістити за зміщенням 0022H.

7 У масиві M(N) однобайтових чисел, розміщених у пам’яті, починаючи з адреси 7000:0010H, підрахувати кількість непарних чисел. Результат розмістити за зміщенням 0030H.

8 У масиві M(N) двобайтових чисел, розміщених у пам’яті, починаючи з адреси 7000:0010H, підрахувати кількість парних чисел. Результат розмістити за зміщенням 0026H.

Контрольні запитання підвищеної складності:

1 Знайти суму елементів головної діагоналі квадратної матриці N×N і розмістити результат за зміщенням 0020H. Початкова адреса двовимірного масиву 7000:0010.

2 Скільки разів буде повторено цикл, якщо у лічильних циклів CL буде спочатку занесено число 2?

10 ПРОГРАМНА РЕАЛІЗАЦІЯ ВУЗЛІВ ТЕЛЕКОМУНІКАЦІЙНОГО ОБЛАДНАННЯ МОВОЮ АСЕМБЛЕР-86

10.1 Способи реалізації алгоритмів

Вхідний контроль:

1 Що таке алгоритм?

2 Які засоби задання алгоритмів Ви знаєте?

Алгоритм будь-якого процесу можна реалізувати трьома основними способами – **апаратним**, **матричним** та **програмним**.

Апаратний спосіб реалізації алгоритмів роботи цифрових пристроїв, на яких будуються вузли телекомунікації, був до недавнього часу домінуючим. Основними його рисами є детермінізм реалізації: навіть незначні зміни в алгоритмі призводять до необхідності перероблення хоча б частини схеми, а також схемотехнічна надмірність: послідовні процеси, які реалізують алгоритм, можуть виконуватись паралельно. Апаратний спосіб забезпечує найвищу швидкість роботи цифрових пристроїв.

Матричний спосіб по суті є також детермінованим, він реалізується за матричною технологією на кришталі мікросхеми. Логічні функції програмованої логічної матриці (ПЛМ, PAL – Programming Array Logic) описуються у вигляді функцій алгебри логіки – булевих рівнянь. Ці рівняння транслюються у карту плавких перемичок спеціалізованим компілятором на комп'ютері, результуюча карта виводиться у вигляді стандартизованого файла. Цей файл завантажується у програматор, і плавкі перемички перепалюються. До переваг ПЛМ відносять програмованість, ПЛМ відносно дешеві порівняно з апаратно реалізованими вузлами. До недоліків ПЛМ слід віднести відсутність у них пристроїв пам'яті та низьку швидкодію. ПЛМ використовуються для реалізації інтерфейсів, контролерів магістралей у мікропроцесорних системах, декодерів адрес тощо.

Програмний спосіб базується на поетапно-послідовній реалізації алгоритмів зі зберіганням проміжних результатів при програмному керуванні усіма виконуваними діями. Такий спосіб є найбільш універсальним і об'єднує можливості оптимального використання апаратних та програмних засобів. Програмний спосіб реалізації вузлів будь-якої апаратури базується на використанні програмованих ВІС-мікропроцесорів та мікроконтролерів і поєднує їх великі можливості за обробки інформації, досягану високу швидкодію, низьку вартість. Цей спосіб є найбільш використовуваним у сучасних реалізаціях вузлів телекомунікацій на апаратно-програмних комплексах.

Програмні засоби можуть замінити апаратні, якщо ця заміна задовольняє вимогам до швидкодії обчислювальної або мікропроцесорної системи і ця заміна економічно доцільна, а також якщо програмна модель реалізує повністю

функції апаратури, а саме сприймання, зберігання, оброблення та видавання даних.

У загальному випадку програмна модель апаратних засобів вміщує програму роботи та значення вхідних змінних (сигналів), які програма переробляє у набори вихідних сигналів.

Контрольні запитання:

1 Чи можна реалізувати матричний спосіб задання алгоритмів на ПЛІС?

2 В якому вигляді повинен бути заданий алгоритм для реалізації на ПЛІС?

3 Чи є можливість реалізувати невироджені цифрові автомати на ПЛІС?

4 Які способи підвищення швидкодії реалізації алгоритмів, заданих програмно, Ви знаєте?

Контрольні запитання підвищеної складності:

1 Які елементи та зв'язки між ними треба змінити у схемі на рис. 3.8 і 3.11, щоб реалізувати нову відповідність коду вихідного сигналу коду МК?

2 Як можна апаратним способом реалізувати розгалуження?

3 Як за допомогою ПЛМ реалізувати цикли?

10.2 Розробка апаратно-програмних комплексів

Вхідний контроль:

1 Які вузли телекомунікацій Ви знаєте?

2 Яку роль у пристроях телекомунікацій відіграють системи синхронізації та генератори імпульсів?

Розробка програмного забезпечення є застосування принципів, навичок та творчого підходу до складання програмних комплексів. Середовищем цих програм є обчислювальна система, на якій вони використовуються. До характерних рис програмного забезпечення можна віднести:

– програми виконуються під єдиним керуванням, частіш за все під керуванням операційної системи, яка працює на центральному процесорі;

– взаємне керування окремими програмами комплексу є підлеглим по відношенню до цього єдиного керування, яке реалізується на більш високому рівні пріоритету та рівнях привілеїв;

– внутрішні рівні операційної системи є сховані від користувачів обчислювального комплексу і не можуть бути змінені користувачем.

При проектуванні програмного комплексу створюється його повний та точний опис, який є комплектом підописів програмних та апаратних компонентів та їх взаємодій. Повний опис програмного компонента – це програма на визначеній мові програмування, семантика якої повинна бути відома до початку програмування.

Розробники програмного забезпечення повинні досягти чотирьох цілей: заданого функціонування, правильності, продуктивності та надійності.

Задане функціонування – це отримання потрібного та правильного результату при введенні заданих вхідних даних. Вхід – це вся інформація, яка сприймається програмним комплексом. Вихід – це видавана інформація. Правильність визначає, наскільки точно програмний комплекс відповідає поставленим цілям. Під продуктивністю розуміють швидкість та ефективність програмного комплексу при споживанні ним ресурсів у процесі досягнення мети. Надійність – це здатність програмного комплексу вірно виконувати свої функції, не зважаючи на відмови компонентів обчислювальної системи. Під відмовою розуміють тимчасову або постійну зміну характеристик компонентів, які призводять до порушення їх функцій.

Програмне забезпечення розробляється за модульним принципом, використовує стандартизовані елементи та обмежені розміри заради гнучкості та універсальності, і компонується шляхом об'єднання модулів.

Мікропроцесорні системи часто вбудовують у вузли систем керування, вимірювальних систем, вузлів обладнання різного призначення, в тому числі телекомунікаційного. Розробники спеціалізованого програмного забезпечення мікропроцесорних систем повинні перш за все враховувати особливості технологічних пристроїв або процесів, якими керує програма. Розробка МПС здійснюється вузько спрямовано для реалізації конкретного алгоритму, і цей принцип розповсюджується як на апаратну, так і на програмну складові. Апаратна частина складається з визначеної кількості модулів конкретного типу, необхідних для реалізації заданого алгоритму, а розроблений додаток фіксується найчастіше у ПЗП і часто не може з метою реалізації заходів безпеки зчитуватись, верифікуватись та змінюватись у процесі експлуатації. Таким чином, багатофункційність є тільки потенціальною властивістю МПС.

Телекомунікації являють собою апаратно-програмні комплекси реального часу, створення програмного забезпечення для яких вимагає чималого досвіду в багатьох галузях. Розвиток інтелектуальних та мультисервісних мереж відбувається високими темпами, розробляються все нові і нові принципи взаємодії різних вузлів та пристроїв, які входять у ці мережі, розробляються нові технології для узгодження й ефективного їх функціонування. В процесі створення програмного забезпечення необхідно врахувати ряд вимог, що є важливим фактором, який впливає на ефективність готового програмного продукту:

– відмовостійкість – можливість системи продовжувати нормальне функціонування в умовах наявності помилок. Висока вірогідність забезпечення завадостійкості веде до забезпечення необхідної якості обслуговування (QoS);

– супроводжуваність – сучасні системи розробляються з урахуванням їх використання протягом достатньо значного терміну. У цей період неодноразово виникає необхідність в її модифікації, що викликається різними причинами (розширення функціональних можливостей, заміна апаратних елементів на сучасніші, зміна стандартів і правил взаємодії із зовнішнім середовищем тощо);

– керованість процесом розробки – великі системи, як правило, відрізняються значним об'ємом і структурною складністю апаратних засобів, а так само значним набором виконуваних функцій. Все це вимагає розробки

програмного забезпечення великого обсягу. Досягаючи певних меж, складність програмного забезпечення призводить до втрати контролю над його розробкою і вимушує розробників робити помилки, які згодом достатньо важко виправити;

– гнучкість – буває так, що є точно визначена структура апаратних засобів і параметри зовнішнього оточення, і більшість систем дозволяють налаштовуватись на ці характеристики. Для великих систем така структура реалізується за допомогою створення бази даних, що містить усі необхідні параметри. Задана структура може реалізуватись як статично до запуску системи, так і динамічно в процесі її функціонування. Динамічне налаштування ускладнюється тим, що зміни у структурі необхідно робити “по живому”, тобто в даних, які можуть у цей момент використовуватися функціональними процесами, що вимагає узгодження між ними, і процесом налаштування;

– стійкість системи – означає її готовність реагувати на непередбачувану поведінку зовнішнього середовища;

– стабільність – вбудована система може перебувати в одному із двох станів: **активному** і **пасивному**. В **активному** стані здійснюється функціонування елементів (можливо, не всіх) програмного забезпечення системи, і система виконує всі або деякі зі своїх функцій; при цьому повний стан системи описується станом апаратури і станом даних в обчислювальному середовищі. Під час переходу системи в **пасивний** стан відбувається зупинка програмного забезпечення; при цьому не всі використовувані ним дані втрачають свою актуальність, частина їх повинна бути збережена до наступного запуску системи, що примушує зберігати ці дані в пам’яті; як засоби обробки таких даних є системи баз даних, що забезпечують організацію, зберігання і доступ до даних;

– безперервність – для більшості систем безперервне і надійне функціонування є критичним; безперервність функціонування означає неможливість зупинки системи, зв'язану, як правило, з важливістю виконуваних нею процесів;

– паралельність – є характерною властивістю більшості систем; наявність апаратних елементів, що працюють у реальному паралельному режимі, й одночасне виконання декількох функціональних процесів вимагають застосування паралельних обчислень; одна з проблем, пов’язана з розподілом ресурсів в програмно-апаратних системах, полягає в тому, що не завжди можна добитися їх монополізації; зокрема, це торкається даних, що відображають стан апаратних елементів системи; такі дані не можуть бути повністю монополізовані функціональним процесом, оскільки існує апаратний процес, який може зажадати їх асинхронну зміну;

– розподіленість – при проектуванні великих вбудованих систем часто використовується принцип модульності. Це дозволяє “збирати” системи різної конфігурації з типових елементів – модулів, збільшувати гнучкість системи. Ще одною перевагою модульної структури є можливість створення розподіленої архітектури вбудованої системи. При цьому кожен окремий модуль включає свій власний управляючий елемент – вбудований процесор. Усі процесори системи об'єднуються в обчислювальну мережу, що забезпечує обмін

інформацією між ними. Зрозуміло, що багато з перерахованих властивостей характерні для всіх великих програмних систем (наприклад, супроводжуваність і керованість), частка характерна тільки для вбудованих систем. Важливо відзначити, що для вбудованих систем докорінним чином змінюється система пріоритетів, наприклад, стійкість і безперервність потрібно забезпечувати навіть у збиток іншим характеристикам.

Розробка програмних моделей вузлів телекомунікацій здійснюється відповідно до рекомендацій ITU серії Z – Z.100, переважно табличним методом. Табличний метод є вільний від недоліків, пов'язаних з нерозумінням фахівців у процесі розробки програмного забезпечення завдяки високому ступеню абстрактності.

Нижче наведені приклади реалізації деяких простих вузлів телекомунікацій.

Контрольні запитання:

1 Які переваги для реалізації вузлів телекомунікацій має програмний метод?

2 Як Ви розумієте модульність побудови програмного забезпечення?

3 Які вимоги пред'являються до програмного забезпечення, яке реалізує вузли телекомунікацій?

4 Які рекомендації міжнародних організацій регламентують створення моделей вузлів телекомунікацій?

Контрольні запитання підвищеної складності:

1 Наведіть приклад, коли при переході системи у пасивний стан, дані повинні зберігатись до наступного запуску системи?

2 Де, на Ваш погляд, у ПК зберігаються дані про адреси векторів переривань BIOS?

3 Чи знищуються вони при вимкненні комп'ютера?

10.3 Приклади реалізації простих вузлів телекомунікацій

10.3.1 Ініціалізація послідовного асинхронного адаптера RS-232-C

Вхідний контроль:

1 Які адреси мають порти послідовного асинхронного адаптера?

2 Чому для передавання даного 41H (підрозділ 6.2) використовується керувальне слово 1EH?

3 З якою метою у цьому ж прикладі використовується один стоповий біт?

Перше, що повинна зробити програма, яка працює з асинхронним адаптером, – це установити протокол обміну та швидкість передавання даних. Після завантаження операційної системи установлюється швидкість 2400 Бод,

не виконується перевірка на парність, використовується один стоповий біт та восьмибітова довжина передаваного символу.

Для протоколу обміну даними, який застосовується у підрозділі 6.2, керувальне слово становить 1EH. Цей режим забезпечує довжину надсилання у бітах – 7 бітів, два стопових біти та контроль на парність.

Для завдання нового значення швидкості обміну даними потрібно установити старший біт керувального слова в 1 та вивести 80H за адресою 3FBH. Після цього двома послідовними командами виведення потрібно завантажити подільник частоти. Молодший байт подільника завантажується у порт 3FBH, а старший – у порт 3F9H. Виберемо швидкість обміну 1200 Бод, якій відповідає подільник 96D = 60H. Перед початком роботи ініціалізується регістр керування перериваннями (порт 3F9H), якщо у програмі навіть не використовуються переривання від самого адаптера. Якщо переривання не потрібні, у порт записується нуль.

Для дозволу переривань необхідно установити в 1 біти порту керування перериваннями 3F9H, відповідні тим перериванням, які потрібно опрацьовувати. Коли відбулося переривання, програма-опрацьовувач переривання повинна проаналізувати його причину за значенням вмісту порту ідентифікації переривання з адресою 3FAH.

Якщо водночас виникають декілька запитів на переривання, біт 0 регістра ідентифікації буде установлено в 1. У такому разі перед завершенням опрацювань переривань треба знов прочитати регістр ідентифікації переривань і опрацювати наступне переривання. Процес повторюється доти, доки біт 0 регістра ідентифікації переривань не дорівнюватиме 0. На цьому ініціалізація завершується.

Нижче наведено фрагмент програми ініціалізації асинхронного адаптера мовою Асемблер-86.

```
        MOV DX,3FBH   ; Завантаження адреси керувального регістра у DX
        MOV AL,80H    ; Порт керування настроюється на забезпечення
                      ; передавання подільника: D7=1, інші біти дорівнюють
                      ; 0, керувальне слово дорівнює 80H
        OUT  DX,AL    ; Виведення керувального слова
        MOV DX,3F8H   ; Завантаження адреси порту 3F8H
                      ; та
        MOV AL,60H    ; завантаження молодшого байта
        OUT  DX,AL    ; подільника частоти
        MOV DX,3F9H   ; Завантаження адреси порту 3F9H
                      ; та
        MOV AL,00H    ; завантаження старшого байта
        OUT  DX,AL    ; подільника частоти
M3:     MOV AL,1EH    ; Завантаження керувального слова в AL
        MOV DX,3FBH   ; Завантаження адреси керувального регістра
        OUT  DX,AL    ; Виведення керувального слова
        MOV AL,00H    ; Заборона переривань від RS-232-C
```

```
        MOV DX,3F9H   ;Завантаження адреси керувального регістра переривань
        OUT  DX,AL    ; Виведення керувального слова
```

10.3.2 Фрагмент програми передавання даних через асинхронний адаптер RS-232-C

```
M1:  MOV DX,3FDH   ; Завантаження адреси регістра стану лінії
     IN   AL,DX    ; Введення слова стану лінії
     AND  AL,20H   ; Регістр зберігання даних є
     JZ   M1       ; порожній, D5=1?
     MOV  AL,41H   ; Так, завантаження даного в AL
     MOV  DX,3F8H  ; Завантаження адреси порту передавання даних
     OUT  DX,AL    ; Виведення даних
     NOP
```

10.3.3 Фрагмент програми приймання даних через асинхронний адаптер RS-232-C

```
     MOV DX,3FDH   ; Завантаження адреси порту стану лінії
M2:  IN   AL,DX    ; Введення слова стану лінії
     AND  AL,01H   ; Перевірка наявності даних в лінії (D0=1?)
     JZ   M2       ; у циклі
     MOV  DX,3F8H  ; Завантаження адреси порту приймання даних 3F8H
                   ; у DX
M5:  IN   AL,DX    ; Введення даних
     NOP
```

10.3.4 Приклад програми ініціалізації RS-232-C та введення-виведення даних, написаної у програмному середовищі TURBO ASSEMBLER (TASM)

```
; Програма ініціалізації послідовного адаптера та введення-виведення
; даних TITLE RS232.ASM

MODEL SMALL
STACK 256
DATA SEGMENT
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA, ES:DATA

START:  MOV   AX,DATA  ; Суміщення
        MOV   DS,AX    ; сегментів
        MOV   ES,AX    ; даних
        MOV   DX,3FBH  ; Завдання
```

```
            MOV     AL,80H    ;
            OUT     DX,AL     ; подільника
            MOV     DX,3F8H   ;
            MOV     AL,60H    ;
            OUT     DX,AL     ; частоти
            MOV     DX,3F9H   ;
            MOV     AL,00H    ; генератора
            OUT     DX,AL     ;
                              ; Завдання нового режиму адаптера
CONTR:      MOV     AL,1EH    ; Завантаження
            MOV     DX,3FBH   ; керувального
            OUT     DX,AL     ; слова (1EH)
INTER:      MOV     AL,00H    ; Заборона
            MOV     DX,3F9H   ; переривань
            OUT     DX,AL     ; від RS-232-C
                              ; Передавання даних
STATUS:     MOV     DX,3FDH   ; Перевірка
            IN      AL,DX     ; стану
            AND     AL,20H    ; буфера передавання
            JZ      STATUS    ; у циклі
SEND:       MOV     AL,41H    ; Передавання
            MOV     DX,3F8H   ; даного
            OUT     DX,AL     ; 41H
                              ; Приймання даних
            MOV     DX,3FDH   ; Перевірка
READY:      IN      AL,DX     ; наявності
            AND     AL,01H    ; даних у лінії
            JZ      READY     ; у циклі
            MOV     DX,3F8H   ; приймання
            IN      AL,DX     ; даного
            JMP     STATUS    ; Циклування програми
            NOP
EXIT:       MOV     AH,4CH    ; Вихід
            INT     21H       ; з програми
        CODE ENDS
        END
```

Циклування програми командою JMP STATUS здійснюється з метою забезпечення перегляду часових діаграм за допомогою осцилографа на перемкнутих контактах 2 та 3 з'єднувача DB9P COM 1. Часові діаграми для наведеного прикладу будуть співпадати з наведеними на рис. 6.2.

Контрольні запитання:

1 З якою метою програмно ініціалізується асинхронний адаптер RS-232-C?

2 З якою метою адреси портів послідовного асинхронного адаптера завантажуються у регістр DX?

3 З якою метою у фрагменті програми у підрозділі 10.3.2 виконується команда AND AL,20H?

4 За якої умови у фрагменти програми у підрозділі 10.3.3 після команди JZ M2 буде виконуватись команда MOV DX,3F8H?

5 Як розподіляється пам'ять по сегментах після виконання директиви MODEL SMALL?

Контрольні запитання підвищеної складності:

1 Як у програмі, написаній у середовищі TASM, завантажуються регістри CS, DS, SS, ES?

2 З якою метою у цій програмі використовується фрагмент:

```
MOV     AH,4CH
INT     21H
```

3 Яку адресу має у таблиці переривань комірка, в якій зберігається адреса підпрограми оброблення INT 21H?

10.3.5 Програмна реалізація генератора імпульсних послідовностей

Вхідний контроль:

1 Генератор, який синхронізує роботу МПС, видає імпульси на МП з частотою 10 МГц; яка тривалість тактового імпульсу МП?

2 Як можна реалізувати програмно часову затримку на 10 мкс?

В основу методу створення програмних моделей генераторів імпульсних послідовностей покладено три прийоми:

– використання різних підпрограм часових затримок;

– використання таблиць в ОЗП, які вміщують інформацію про імпульсні послідовності;

– використання алгоритмів обчислення часових логічних функцій.

Вираз для часової логічної функції генератора імпульсної послідовності $Y = f(t)$, де t – дискретний час, показує залежність вихідного сигналу від дискретного реального часу. Обчислення часової логічної функції повинне відбуватися за рівні інтервали часу, інакше синхронізація формованих послідовностей буде порушуватись. На рис. 10.1 показано модельовану імпульсну послідовність кінцевої довжини, яка виводиться через молодший розряд паралельного порту PORT.

Імпульсна послідовність розбивається на інтервали Δt та кодується, закодований опис завантажується у пам'ять.

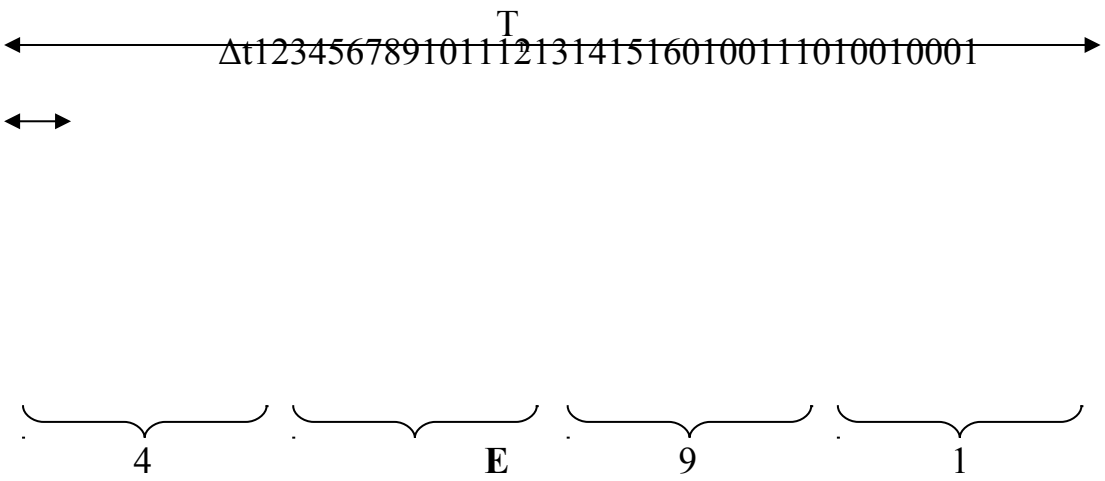


Рисунок 10.1 – Кодування імпульсної послідовності

Наведена нижче підпрограма ІМ реалізації імпульсної послідовності працює таким чином: через рівні інтервали часу Δt наступний байт опису послідовності виводиться у PORT. Інтервал Δt визначається кількістю тактів синхронізації МП, які потрібні для виводу байта у порт (10 тактів = 1 мкс) та кодом часу затримки у підпрограмі DELAY. Таким чином $\Delta t = 1_{\text{МКС}} + t_{DELAY}$. Підпрограма моделює тільки період імпульсної послідовності, для моделювання довгої серії імпульсів треба організувати цикл звернення до підпрограми ІМ:

```
IM: MOV  BX,A1ABH  ; Занесення до BX адреси таблиці опису
M2: MOV  CL,02H    ; Занесення до CL кількості байтів опису
    MOV  DH,8H     ; Занесення до DH кількості бітів у байті
    MOV  AL,[BX]   ; Занесення до AL чергового байта опису
    MOV  DL,AL     ; Запам’ятовування цього байту у DL
M1: AND  AL,01H    ; Виділення молодшого біта
    OUT  PORT      ; Вивід його у PORT
    SHR  DL,1      ; Логічний зсув праворуч на 1 розряд
    MOV  AL,DL     ; Запам’ятовування нового значення байта
    CALL DELAY     ; Затримка сигналу на виході порту
    DEC  DH        ; Декрементування лічильника бітів
    JNZ  M1        ; Повернення на початок циклу
    INC  BX        ; Нарощування адреси елемента таблиці
    LOOP M2        ; Звернення до наступного елемента таблиці
    RET            ; Повернення з підпрограми
    NOP
```

10.3.6 Програмне вимірювання періоду імпульсної послідовності DET

Період імпульсної послідовності $T_{\text{пер}}$ вимірюється підрахуванням кількості відомих малих інтервалів часу Δt, які накопичуються за один період імпульсної послідовності. Імпульсна послідовність показана на рис. 10.1.

Частота імпульсів повинна знаходитись у межах 5 Гц ... 167 кГц. Підпрограма DET кожні $\Delta t = 3$ мкс опитує порт, на молодший розряд якого надходить послідовність імпульсів. Фіксується фронт імпульсу, після чого у лічильнику CX накопичується кількість дискретних інтервалів часу ТІ, протягом яких у порт надходить високий рівень напруги, а у лічильнику BX накопичується кількість інтервалів часу ТП, протягом яких зберігається низький рівень напруги. Числа ТІ та ТП запам'ятовуються відповідно за адресами 20H та 22H у сегменті даних, а кількість інтервалів часу за період $T_{пер} = ТІ + ТП$ запам'ятовується у комірці пам'яті за адресою 24H у сегменті даних.

В основній програмі після повернення з підпрограми DET можна підрахувати тривалість імпульсу t_i, тривалість паузи t_n та період $T_{пер}$ у секундах

$t_i = \Delta t(ТІ) = 3$ мкс (ТІ);
$t_n = \Delta t(ТП) = 3$ мкс (ТП);
$T_{пер} = \Delta t(ТІ + ТП) = 3$ мкс (ТІ + ТП).

Фрагмент програми DET наведено нижче:

```
      MOV DX, PORT  ; Завантаження до DX адреси порту
DET:  IN AL, DX     ; Фіксація
      AND AL, 01H   ; фронту
      JNZ DET       ; імпульсу
M1:   IN AL, DX
      AND AL, 01H
      JNZ M1
      MOV CX, 00H   ; Скидання лічильника часу (ТІ) імпульса
M2:   INC CX        ; Вимірювання ТІ
      IN AL, DX
      AND AL, 01H
      JNZ M2
      MOV [20H], CX ; Занесення ТІ у пам'ять
      MOV BX, 00H   ; Скидання лічильника часу (ТП) паузи
M3:   INC BX        ; Вимірювання ТП
      IN AL, DX
      AND AL, 01H
      JNZ M3
      MOV [22H], BX ; Занесення ТП у пам'ять
      ADD BX, CX    ; Визначення періоду послідовності
                    ; імпульсів (Tпер)
      MOV [24H], BX ; Занесення Tпер у пам'ять
      RET           ; Повернення з підпрограми
```

Контрольні запитання:

1 Яка підпрограма використовується для формування будь-яких часових послідовностей?

2 Як за допомогою табличного метода програмно реалізувати генератор імпульсних послідовностей?

3 Як, використовуючи комп'ютер або МПС, здійснити вимірювання періоду імпульсної послідовності?

Контрольні запитання підвищеної складності:

1 Напишіть програму формування імпульсної послідовності з коефіцієнтом заповнення 0,5.

2 Виведіть цю імпульсну послідовність через порт RS-232-C.

10.3.7 Програмна реалізація мультиплексора

Вхідний контроль:

1 Напишіть у вигляді таблиці алгоритм роботи мультиплексора з вісім'ю інформаційними входами.

2 Який активний рівень має сигнал дозволу роботи мультиплексора?

Реалізувати програмно мультиплексор з вісім'ю інформаційними входами; залежно від коду на адресних входах один із входів підключається до виходу; сигнал дозволу має нульове активне значення. Нижче наведено фрагмент підпрограми, яка моделює такий мультиплексор.

```
        MOV AH, 04H   ; Завантаження керувального слова у регістр AH
        MOV BH, 63H   ; Завантаження байта даних
        CALL PMS      ; Звернення до підпрограми, яка реалізує
                      ; мультиплексор
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _
PMS:    MOV AL, AH    ; Запам'ятовування керувального слова в AL
        TEST AL, 10H  ; Дозвіл на підключення біта даних до виходу є?
        JNZ PP2       ; Ні, перехід на установлення дозволу
        OR AL, AL     ; Так, адреса є нульова?
        JNZ M2        ; Ні, перехід на оброблення адреси
        RCR BH, 1H    ; Так, запам'ятовування молодшого біта даних
                      ; у CF
        RCL AL, 1H    ; Перенесення молодшого біта даних у нульовий
                      ; розряд AL
        JMP EXIT      ; Перехід на повернення з підпрограми
M2:     RCR AL, 1H    ; Оброблення вказаної
        AND AL, 07H   ; адреси
        MOV BL, AL    ; Запам'ятовування адреси у лічильнику адрес
        RCR BH, 1H    ; Виключення з розряду нульового біта даних
PP1:    RCR BH, 1H    ; Заміщення CF бітами даних, починаючи
```

```
                    ; з першого у циклі
        DEC BL      ; Зменшення адреси до нуля
        JNZ PP1     ; у циклі
        RCL AL, 1H  ; Внесення адресованого біта даних
                    ; у молодший розряд AL
M3:     MOV AH, AL  ; Повернення керувального слова з адресованим
                    ; бітом даних у AH
        JMP EXIT    ; Безумовний перехід на повернення
                    ; з підпрограми
PP2:    AND AH, EFH ; Установлення дозволу на підключення біта
                    ; даних до виходу при збереженні
                    ; керувального слова
EXIT:   RET
```

Вісім інформаційних входів мультиплексора моделюються регістром BH. У нульовому розряді регістра AL будемо отримувати результат – прямий вихід мультиплексора. У розряди 1, 2, 3 регістра AL заноситься код адреси біта даних, а розряд 4 моделює вхід дозволу. Код адреси зменшується у циклі при одночасній фіксації в ознаці CF значення біта даних в обраному розряді.

Контрольні запитання:

1 Чому у програмі, яка зреалізовує мультиплексор, окремо розглядається випадок, коли код адреси дорівнює нулю?

2 Яким способом, апаратним або програмним, доцільніше реалізувати мультиплексор на 32 інформаційних входи і чому?

Контрольні запитання підвищеної складності:

1 Як треба змінити фрагмент програми реалізації мультиплексора, щоб перевіряти наявність дозволу на роботу мультиплексора поза підпрограмою PMS?

2 Як у програмі здійснюється синхронізація адреси й значення біта даних в обраному розряді?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ ДО 2-го МОДУЛЯ

1 Брэй Б. Микропроцессоры Intel: 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro Processor, Pentium II, Pentium III, Pentium 4. Архитектура, программирование и интерфейсы. – 6-е изд. / Пер. с англ. – С.Пб.: БХВ – Петербург, 2005. – 1328 с.: ил.

2 Юров В., Дорошенко С. Assembler: Учебный курс. – С.Пб.; Изд. «Питер», 1999. – 672 с.: ил.

3 Майко Г. В. Ассемблер для IBM PC. – М.: Бизнес-Информ, Сирин, 1997. – 250 с.: ил.

4 Митрофанов Ю. М., Ошаровська О. В., Хіхловська І. В. Програмування на мові Асемблер: Навчальний посібник для самостійної роботи з курсу «Цифрова техніка та мікропроцесори». – Одеса: УДАЗ, 1997. – 25 с.: іл.

5 Брамм П., Брамм Д. Микропроцессор 80386 и его программирование. Пер. с англ. – М.: Мир, 1990. – 448 с.: ил.

6 Майоров В. Г., Гаврилов А. И. Практический курс программирования микропроцессорных систем. – М.: Машиностроение. 1989. – 272 с.: ил.

7 Григорьев В. Л. Программирование однокристальных микропроцессоров. – М.: Энергоатомиздат, 1987. – 288 с.: ил.

8 Абель П. Язык Ассемблер для IBM PC и программирования. – М.: Высшая школа, 1992. – 447 с.: ил.

9 Лю Чжен–Ю, Гибсон Г. Микропроцессоры семейства 8080/8088. – М.: Радио и связь, 1987. – 512 с.: ил.

10 Микропроцессорный комплект К1810: Структура, программирование, применение: Справочная книга / Ю. М. Казаринов, В. Н. Номоконов, Г. С. Подклетнов, Ф. В. Филиппов; Под ред. Ю. М. Казаринова. – М.: Высшая школа, 1990. – 269 с.: ил.

11 INTERNATIONAL TELECOMMUNICATION UNION. TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU. Addendum 1 (10/96). SERIES Z: PROGRAMMING LANGUAGES. Specification and Description Language (SDL).