МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Волинський національний університет імені Лесі Українки
Кафедра комп'ютерних наук та кібербезпеки

В. В. Булатецький, Л. В. Булатецька

ЗАГАЛЬНІ ПРИНЦИПИ ФУНКЦІОНУВАННЯ ТЕХНІЧНИХ ЗАСОБІВ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

Текст лекцій
нормативної навчальної дисципліни
"Архітектура обчислювальних систем"

Луцьк 2021

УДК 004.655

Рекомендовано до друку науково-методичною радою
Волинського національного університету імені Лесі Українки
(протокол №8 від 22 квітня 2021 р.)

Рецензенти:
Собчук О. М. – кандидат пед. наук, доцент кафедри загальної математики та методики навчання інформатики Волинського національного університету імені Лесі Українки
Костючко С. М. – кандидат технічних наук, доцент кафедри комп'ютерної інженерії та кібербезпеки Луцького національного технічного університету

Загальні принципи функціонування технічних засобів обчислювальних систем: *текст лекцій нормативної навчальної дисципліни "Архітектура обчислювальних систем"*. укладачі *Булатецький Віталій Вікторович, Булатецька Леся Віталіївна.* Луцьк : ВНУ імені Лесі Українки, 2021. 57 с.

Текст лекцій призначений для студентів напряму підготовки 122 Комп'ютерні науки, 014 Середня освіта (Інформатика) та є логічним доповненням до лекційного курсу нормативної навчальної дисципліни "Архітектура обчислювальних систем". Призначений для оволодіння теоретичними та практичними навиками функціонування технічних засобів обчислювальних систем. Рекомендовано викладачам та студентам які викладають або вивчають архітектуру обчислювальних систем у вищих навчальних закладах.

ЗМІСТ

Лекція 1
ОСНОВНІ ВІДОМОСТІ З ТЕОРІЇ ІНФОРМАЦІЇ ТА ТЕОРІЇ АЛГОРИТМІВ. ОБЧИСЛЕННЯ ТА ТЕХНІЧНІ ЗАСОБИ ЇХ ЗДІЙСНЕННЯ

План

1.1. Поняття інформації
1.2. Джерела інформації
1.3. Подання та вимір кількості інформації
1.4. Алгоритми арифметичних та логічних операцій
1.5. Апаратні засоби алгоритмічно універсальної обчислювальної системи

1.1. Поняття інформації.

Сучасна цифрова обчислювальна техніка – найбільш потужний засіб автоматичного опрацювання інформації, який втілює в собі досягнення багатьох галузей науки і техніки і є результатом росту усвідомленої суспільної необхідності в підвищенні продуктивності праці. При цьому *цифровими обчислювальними системами* прийнято називати комплекси технічних засобів, що відповідно до деякої програми, реалізують математичні операції по опрацюванню інформації, поданої в цифровій формі.

Поняття інформації є одним із найбільш загальних понять науки і позначає сукупність деяких зведень, даних, знань і т. п.

В обчислювальній техніці *інформація – властивість об'єктів і явищ матеріального світу породжувати різноманіття станів, що за допомогою відображення, передаються від одного об'єкта до іншого і запам'ятовуються в його структурі (іноді у зміненому виді)*. При цьому під інформацією розуміють не самі об'єкти і явища, а їх істотні і представницькі ознаки, їхнє відображення у вигляді чисел, формул, описів, креслень, символів, зразків і інших абстрактних характеристик. Сама по собі інформація може бути віднесена до області абстрактних категорій, подібних, наприклад, до математичних формул.

Передачею інформації називають перенос інформації на відстань. Для цього необхідно як мінімум два об'єкти – *джерело* інформації і *приймач* інформації, з'єднані *каналом зв'язку*. У каналі зв'язку інформація представляється в матеріально-енергетичній формі за допомогою множини станів деяких матеріальних об'єктів, які називаються *носіями інформації* або *сигналами*. Відображення множини станів джерела інформації в множині станів носія, називають *кодуванням*, а відображення множини станів носія інформації в множину станів приймача – *декодуванням*. Засіб відображення інформації, що установлює відповідність між елементами повідомлень і сигналами, називають *кодом*.

1.2. Джерела інформації

Джерела інформації розділяють на *неперервні* і *дискретні*.

Неперервні джерела створюють повідомлення, що являють собою сигнали, які подають які-небудь фізичні величини, що змінюються неперервно і

приймають нескінчене число значень у деякому діапазоні. Таке представлення інформації використовується в *аналогових* обчислювальних і моделюючих пристроях.

Дискретні джерела інформації створюють повідомлення, що складаються з кінцевої множини елементів (букви або символи). Букви відображаються сигналами, що приймають кінцеве число значень. У сучасній обчислюваній техніці поняття букви містить у собі символи різноманітних систем писемності, цифри, знаки і т. п. Якщо елементам дискретного повідомлення поставлені у відповідність цифри або деякі їх сукупності, то таке представлення інформації називають *цифровим* (числовим). Кінцевий набір букв, який використовують для відображення дискретних повідомлень, називають *алфавітом*. Будь-яку кінцеву послідовність букв деякого алфавіту прийнято називати *словом* (у тому числі послідовність, що складається з цифр). Число букв у слові називають його довжиною.

Кількість інформації, переданої від джерела до приймача, пов'язана з ймовірністю перебування джерела в тому або іншому стані. Якщо стан джерела відомий заздалегідь (до передачі інформації), то кількість інформації, одержуваної приймачем при передачі, дорівнює нулю. Якщо ж стан джерела не відомий заздалегідь, то кількість одержуваної інформації визначається формулою:

$$l = -\sum_{i=1}^{N} P_i \log_k P_i, \qquad (1)$$

де N – число станів, у яких може знаходитися джерело, P_i – ймовірність появи i-го стану ($i = 1, 2, \ldots, N-1, N$). При рівноймовірних станах джерела (тобто при $P_1 = P_2 = \ldots = P_N = 1/N$)

$$l = \log_k N. \qquad (2)$$

Основа логарифму k з формули 2 визначає одиницю кількості інформації. При $k = 2$ відповідна одиниця називається *біт* (*Bit* – від слів *binary digit*). Така одиниця частіше усього зустрічається в техніці, що обумовлено найбільш частим використанням двозначного алфавіту для подання дискретної інформації. *Один біт дорівнює кількості інформації, одержуваної від джерела з двома рівноймовірними станами.*

1.3. Подання та вимір кількості інформації

Сучасні обчислювальні системи можуть опрацьовувати не тільки числову інформацію, але й інформацію, задану будь-якими іншими символами. Звичайно, для подання одного символу служить слово довжиною в $2^3 = 8$ біт, що одержало назву *байту*. За допомогою слів такої довжини можна закодувати $2^8 = 256$ різноманітних символів, чого цілком достатньо при розв'язування багатьох задач, пов'язаних з опрацюванням символьної інформації. Кількість інформації в цьому випадку зручно вимірювати також у байтах.

Для виміру великих обсягів інформації в обчислювальній техніці застосовують спеціальні одиниці, що позначаються К і М і читаються відповідно «кіло» і «мега». При цьому 1К = 1024 = 2^{10}, 1М = = 1 048 576 = 2^{20}. Наприклад, 1Мбайт = 2^{10}Кбайт = 2^{20}байт = 2^{23} біт, 1Мбіт = 2^{10}Кбіт = 2^{20} біт, 1Кбайт = 2^{10}байт = 2^{13}біт, 1Кбіт = 2^{10}біт. Іноді в наближених розрахунках допускають, що $К \approx 10^3$, $М \approx 10^6$.

Одиницею виміру швидкості передачі інформації по каналах зв'язку слугує *бод*, рівний 1 біт/с.

Крім перерахованих одиниць для виміру кількості інформації, обробленої і збереженої в обчислювальнійситемі, використовують також одиниці, що не мають постійного кількісного еквіваленту. До таких одиниць відноситься *поле, слово, масив, сегмент* і інші. *Поле* являє собою групу біт, що має певне значення (наприклад, поле, у якому вказується в кодованому виді операція, виконувана на обчислювальній системі).

Сукупність біта, байтів, полів, слів, що об'єднуються деякою загальною ознакою (наприклад, вихідні дані для розв'язку задачі), називають *масивом*. *Сегмент* – упорядкована сукупність біта, байтів, полів, слів, масивів, згрупованих разом із метою найменування.

Обробка інформації на обчислювальній системі полягає у виконанні ряду операцій відповідно до деякого алгоритму, що в результаті призводить до одержання результату або розв'язку. Поняття алгоритму не має точного математичного визначення і його зміст абстрагується з досвіду (так само, як, наприклад, зміст понять «множина», «число», «відповідність» і ін.). Тому усі відомі визначення алгоритму є в тій або іншій мірі неповними. Проте для зручності усе ж можна вважати, що *алгоритм* – це спосіб перетворення інформації, що задається за допомогою кінцевої системи правил. Таке визначення є достатньо загальним і водночас найбільше відповідним цільовому призначенню підручника і специфіці матеріалу, що викладається.

1.4. Алгоритми арифметичних та логічних операцій

Аналіз елементарних операцій по опрацюванню інформації, що зустрічаються в реальних алгоритмах, показує, що їх можна розділити на дві групи: *арифметичні і логічні. Арифметичні* операції виконують безпосереднє перетворення інформації, а *логічні* визначають напрямок процесу опрацювання інформації. У алгоритмах, арифметичні та логічні операції чергуються у визначеній послідовності. Якщо виконання алгоритму зводиться до арифметичних операцій, то такий алгоритм називають чисельним.

Два алгоритми вважаються *рівними*, якщо для деякого перетворення інформації вони встановлюють однакову відповідність між вхідними і вихідними словами і збігаються системи правил, що задають ці алгоритми. Два алгоритми називаються *еквівалентними*, якщо вони встановлюють однакову відповідність між вхідними і вихідними словами, але відрізняються засоби їхнього задання.

Алгоритми мають властивості *визначеності, масовості і результативності*. Властивість *визначеності* виражає той факт, що сукупність операцій, виконуваних відповідно до деякого алгоритму, не припускає ніякої вільності що-

до їхньої послідовності та тлумачення, тобто є детермінованим процесом. *Масовість* алгоритму означає можливість рішення з його допомогою цілого класу задач із змінними вихідними даними. *Результативність* алгоритму полягає в тому, що шуканий результат може бути отриманий за допомогою алгоритму шляхом виконання кінцевого числа операцій при всіх припустимих значеннях вихідних даних. Розглянуті властивості алгоритму є емпіричними і їх не можна вважати визначенням поняття алгоритм.

Областю придатності алгоритму називають найбільшу область вихідних даних, на якій алгоритм має властивість результативності. Якщо вихідні дані не входять в область придатності алгоритму, то він не забезпечує одержання результату за кінцеве число операцій.

Алгоритмічною системою прийнято називати загальний стандартний спосіб задання алгоритмів. У теорії і проектуванні технічних засобів обчислювальної техніки для цих цілей використовують дві алгоритмічні системи: операторні описи (ОО) і граф-схеми алгоритмів (ГСА) У ОО буквами позначають окремі дії алгоритмів по переробці інформації – операції та логічні умови, що перевіряються.

Послідовне виконання кількох операцій позначається як їх *добуток*, причому ліва операція виконується раніше правої. У такому лінійному записі алгоритму операція відрізняється від логічної умови тим, що після останньої ставлять стрілку, спрямовану вверх і позначену числовим індексом. Якщо логічна умова A виконана (тобто $A = 1$), то здійснюється перехід до операції або логічної умови, що вказана стрілкою. Якщо ж умова A не виконана $(A == 0)$, то здійснюється перехід до операції або логічної умови, записаної безпосередньо за умовою A. Наприклад, операторний опис:

$$B_1 B_2 \downarrow {}^{2}\, B_3 A_1 \uparrow {}^{1}\, B_4 B_5 \downarrow {}^{1}\, A_2 \uparrow {}^{2}\, B_6 \qquad (3)$$

означає, що після виконання операцій B_1, B_2 і B_3 необхідно перевірити логічну умову A_1. Якщо $A_1 = 0$, то далі необхідно переходити до операцій B_4, B_5, і логічної умови A_2. Якщо ж $A_1 = 1$, то варто відразу перейти до перевірки A_2. У залежності від значення A_2 можливі два варіанти продовження алгоритму: виконати операцію B_6 (при $A_2 = 0$) або B_3 і далі перевірити A_1 (при $A_2 = 1$).

При записі алгоритмів у виді ГСА використовують три основних символи, що позначені на рис1. і називаються вершинами ГСА. Початок і кінець алгоритмів позначають вершиною (рис. 1*а*). Будь-який алгоритм, поданий ГСА, починається і закінчується цією вершиною. Операторною вершиною (рис. 1*б*) ГСА називають умовне або змістовне позначення виконуваної операції.

Змістовне позначення операції зручно записувати за допомогою оператора присвоювання, що позначається як :=. Наприклад, запис $X := X+1$ означає, що до змінній розміру X необхідно додати 1 (тобто змінній X «присвоїти» нове значення, рівне $X + 1$). Сукупність операторів присвоювання, записаних в одній і тій же операторній вершині, виконується одночасно. Умовна вершина (рис. 1*в*) відповідає логічній умові, що перевіряється, два можливих виходи якого позначені 0 (умова не виконана) і 1 (умова виконана). Усередині умовної вершини

записують або логічний вираз, що відбиває зміст умови, що перевіряється, або двійкову змінну (наприклад, A), що позначає результат перевірки. Лінії на ГСА означають переходи від однієї вершини до іншої.

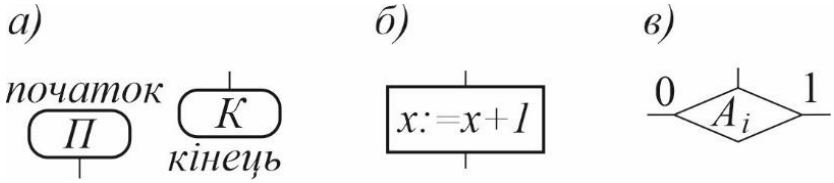


Рис. 1. Позначення вершин при записі алгоритмів у вигляді ГСА

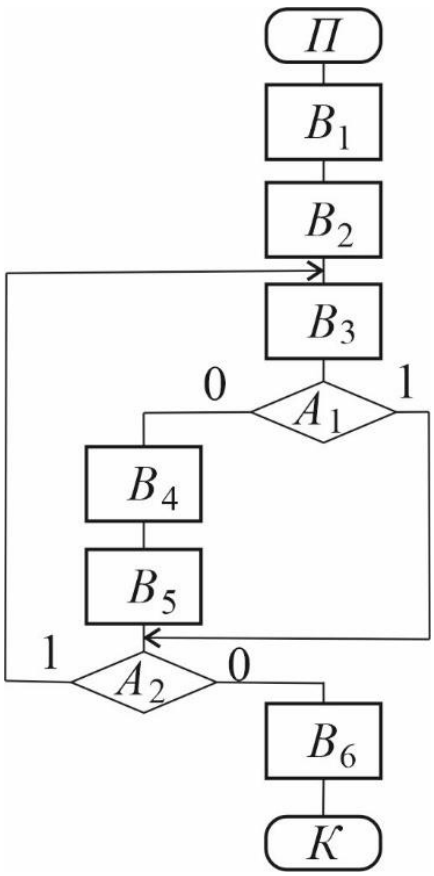


Рис. 2. Граф-схема алгоритму поданою формулою 3

На рис. 2, як приклад побудована граф-схема алгоритму по приведеному операторному описі (формула 3).

1.5. Апаратні засоби алгоритмічно універсальної обчислювальної системи

Головною відмінною ознакою цифрових обчислювальних систем є те, що в них автоматизований процес обчислень за рахунок використання принципу програмного керування (основні ідеї програмного керування викладені англійським математиком Ч. Беббиджем у 1833 р.) і принципу збереженої в пам'яті обчислювальної системи програми обчислень (сформульований у 1945 р. аме-

риканським ученим Дж. фон Нейманом і незалежно від нього в 1950 р. радянським ученим академіком С. А. Лебедєвим).

Інформація, що обробляється обчислювальною системою, відображається у вигляді сукупності цифр (чисел) у деякій системі числення, самі ж цифри відображаються сигналами, що мають кінцеве число рівнів квантування (частіше усього два). Перед початком опрацювання інформації алгоритм опрацювання повинен бути записаний як послідовність тих арифметичних і логічних операцій, для виконання яких у складі обчислювальної системи є відповідні засоби. Такий запис називають *програмою*. Будь-яка програма складається з окремих *команд*, кожна з яких визначає дії обчислювальної ситеми по виконанню якоїсь однієї операції. Всі *операції* в обчислювальній системі реалізуються за допомогою *апаратних (технічних)* або *програмних засобів*. При цьому під *апаратними засобами* розуміють комплекс технічних пристроїв (звичайно електронних), внутрішня структура котрих, а також зв'язки між ними побудовані таким чином, щоб забезпечити реалізацію заданих операцій. *Програмні засоби* – це програми виконання заданих операцій, як послідовностей деяких найпростіших (елементарних) операцій, реалізованих, у свою чергу, апаратними засобами.

Сутність принципу програмного керування полягає в тому, що процес опрацювання інформації здійснюється на основі інформації, заданої для керування цим процесом. На перших обчислювальних системах програма заносилася на перфострічку. Машина послідовно зчитувала з перфострічки і розшифровувала інформацію про керування процесом обчислень. Ця ж ідея, але на якісно інший технічній основі, реалізується у всіх сучасних програмно керованих обчислювальних систем.

Множина всіх апаратних та програмно реалізованих операцій в обчислювальній системі складає її *операційні ресурси*. Обчислювальні системи, операційні ресурси яких забезпечують принципову можливість виконання будь-якого алгоритму опрацювання інформації, прийнято називати *алгоритмічно універсальними*. Для алгоритмічної універсальності обчислювальної системи достатньо наявності в її операційних ресурсах лише чотирьох операцій: пересилки слова з будь-якої комірки пам'яті в будь-яку іншу комірку, додавання і віднімання одиниці до слова, умовного переходу по співпадінню слів і безумовної зупинки обчислювальної системи. Проте лише в деяких найпростіших мікропроцесорах набори операцій близькі до мінімальних. У переважній більшості обчислювальних систем і мікропроцесорів операційні ресурси значно повніші і складаються вони з десятків і сотень операцій.

Апаратні засоби будь-якої алгоритмічно універсальної обчислювальної системи можна розділити на три основні частини – *пам'ять, процесор і периферійні пристрої* (рис. 3), причому, число пристроїв пам'яті та процесорів у конкретних обчислювальних систем може варіювати від одиниці до кількох десятків, а периферійних пристроїв – до декількох сотень. Пам'ять обчислювальної системи служить для збереження вихідних даних, програм опрацювання інформації, проміжних і остаточних результатів.

У сучасних великих універсальних обчислювальних системах пам'ять являє собою складну багаторівневу систему. У цій системі звичайно виділяють рівні *надоперативної, оперативної, буферної і зовнішньої пам'яті*. Кожний на-

ступний рівень відрізняється від попереднього, у першу чергу, найважливішими технічними характеристиками пам'яті – *ємністю і швидкодією. Ємністю* пам'яті називають максимальну кількість інформації, що може бути в ній записано. *Швидкодія* пам'яті характеризується тривалістю операцій *читання* і *запису* – двох основних операцій, виконуваних у пам'яті обчислювальної системи. До складу пам'яті обчислювальної системи можуть входити також і програмні засоби, що забезпечують керування переміщенням інформації з рівнів пам'яті, упорядковане розміщення інформації, проведення спеціальних перевірочних процедур і т. п. Таку пам'ять називають *віртуальною*, або *математичною пам'яттю*. У малих обчислювальних системах і мікро-обчислювальних системах структура пам'яті істотно простіша і включає один, два рівня (наприклад, оперативну, або оперативну і зовнішню).

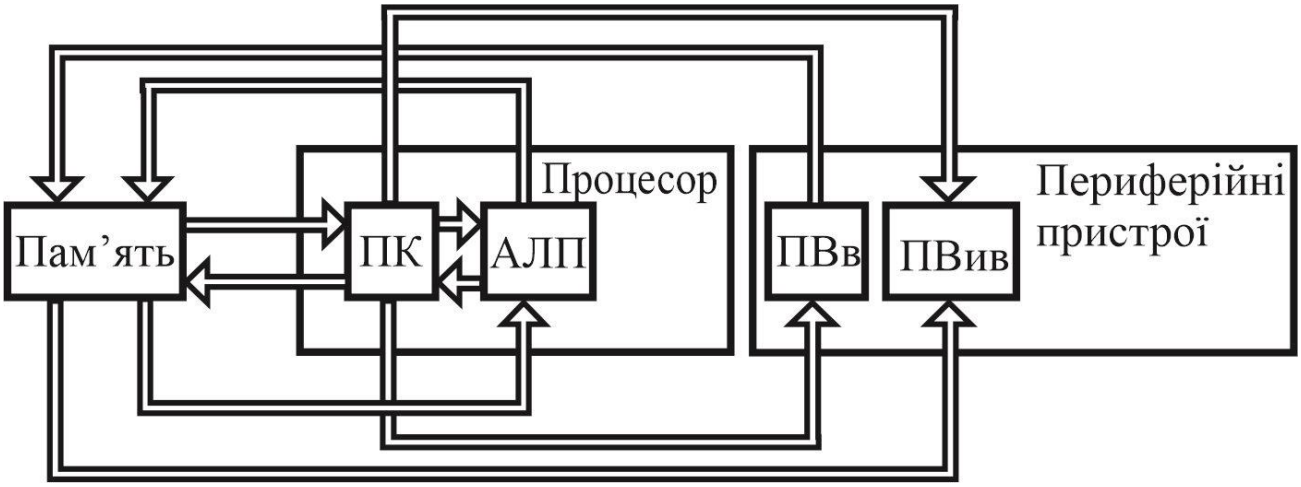


Рис. 3. Апаратні засоби алгоритмічно універсальної обчислювальної системи

У процесорі обчислювальної системи зосереджені всі процеси по опрацюванню інформації. Процесор складається з *арифметично-логічного (операційного)* пристрою й пристрою *керування*. Арифметико-логічним пристроєм (АЛП) називають ту частину процесора, що призначена для виконання арифметичних і логічних операцій над словами, що надходять із пам'яті обчислювальної системи. При цьому слова (числа), над якими виконується деяка операція в АЛП, називають *операндами*. Будь-яке АЛП має у своєму складі кілька *регістрів* і *функціональних (комбінаційних)* схем. *Регістри* призначені для збереження операндів у процесі виконання операцій, а за допомогою *функціональних схем* виконуються необхідні перетворення операндів при передачі їх з одного регістра на інший. Всі операції в АЛП реалізуються як просторово-тимчасові послідовності деяких елементарних операцій (мікрооперацій) над словами, кожна з який є сукупністю операцій над буквами, що складають дані слова.

До числа основних елементарних операцій, що виконуються в АЛП, відносяться:

1) передача (прийом, видача) операнда (слова) на регістр;

2) зсув (арифметичний, циклічний, логічний, модифікований) операнда на задане число розрядів;

3) додавання до слова або віднімання з нього 1 (у більш загальному випадку – деякої константи);

4) порівняння операндів (за принципом «більше - менше - дорівнює»);

5) порозрядні логічні операції диз'юнкції, кон'юнкції, рівнозначності і додавання по модулю, рівному значності алфавіту;

6) підсумовування двох операндів, що відображають числа в одній і тій же системі числення;

7) перетворення кодів операндів, включаючи інверсію, доповнення, дешифрацію й ін.

Перераховані елементарні операції можуть мати декілька варіантів, наприклад, порівняння може виконуватися за принципом «дорівнює - не дорівнює», або як операція виділення більшого операнда.

Пристрій керування (ПК) у складі процесора призначений для розшифровування команд і формування послідовностей керуючих сигналів. Ці сигнали включають у роботу окремі вузли процесора, що в результаті призводить до виконання дій, що вказуються командою. Для виконання однієї елементарної операції в АЛП необхідно подати один сигнал від ПК по ланцюгах керування.

Відомо три основних типи ПК – мікропрограмні, апаратні і змішані.

У *мікропрограмних* ПК кожному керуючому сигналу відповідає визначене слово (частина слова), що зберігається в спеціальній пам'яті і називається *мікрокомандою*. Послідовності керуючих сигналів для деякої операції відповідає сукупність мікрокоманд, що називається *мікропрограмою*. Керування виконанням операції здійснюється шляхом читання з мікропрограмної пам'яті мікрокоманд і їхнє перетворення в сигнали по керуючих ланцюгах. Зміна набору команд, виконуваних процесором, зводиться до заміни вмісту пам'яті мікропрограм і не потребує зміни складу апаратних засобів обчислювальної системи.

У *апаратних* ПК керуючі сигнали формуються спеціальними апаратними засобами (електронними схемами) і зміна раніше закладеної структури послідовностей цих сигналів без змін в апаратних засобах тут неможлива.

У ПК *змішаного* типу керування частиною операцій здійснюється мікропрограмним способом, а іншою частиною – апаратним способом.

Розрізняють також *централізовані* і *децентралізовані* ПК. У централізованих ПК усі керуючі сигнали, необхідні для виконання будь-якої операції, виробляються безпосередньо в ПК. При децентралізованому керуванні ПК складається з центрального ПК (ЦПК) і пристроїв місцевого керування (ПМК).

ЦПК формує тільки основні керуючі сигнали, що звичайно відповідають виду виконуваної операції (наприклад, «множення», «додавання» і т. п.). Ці сигнали впливають на ПМК, що і виробляють сигнали виконання окремих елементарних операцій

Периферійні пристрої призначені для перетворення форми відображення інформації при введенні її в обчислювальну систему і виведення з обчислювальної системи.

Для створення периферійного устаткування використовується дуже широке коло фізичних ефектів і явищ. Принципи побудови і структури периферійних пристроїв у значній мірі залежать від конкретних їхніх застосувань.

Обчислювальні системи, що не є алгоритмічно універсальними, а також обчислювальні системи, призначені для рішення задач одного класу (або навіть одна задача з різними вихідними даними), відносять до класу спеціалізованих обчислювальних систем (проблемно-орієнтованих або обчислювальних систем із жорсткою програмою). Деякі з основних технічних характеристик (продуктивність, вартість, надійність та ін.) таких обчислювальних систем оптимізовані на конкретні застосування. Спеціалізована обчислювальна система має одну програму або набір програм що переключаються, записаних у пам'ять. До класу спеціалізованих обчислювальних систем відносяться і неалгоритмічні обчислювальні системи, у яких обчислювальний процес визначається не послідовністю і сукупністю елементарних операцій у залежності від зовнішньої змінюваної програми, а математичним описом процесу опрацювання інформації, тобто жорсткою програмою, призначеною для побудови внутрішньої структури апаратних засобів і зв'язків між ними.

Два напрямки розвитку засобів обчислювальної техніки – по шляху створення універсальних і спеціалізованих обчислювальних систем не виключають, а взаємно доповнюють один одного. Універсальні обчислювальні системи мають дуже широку область застосування і випускаються значними серіями. Для рішення ж порівняно вузького кола задач більш ефективні спеціалізовані обчислювальні системи. Клас спеціалізованих обчислювальних систем є областю апробації нових методів автоматизації обчислень, що потім впроваджуються в універсальні обчислювальні системи. До числа таких методів, наприклад, ставляться багаторівневе опрацювання інформації з розпаралелюванням обчислювального процесу на окремі незалежні гілки; децентралізація обчислень за допомогою багатопроцесорних систем і аппаратно реалізованих підпрограм (попередніх процесорів); виконання складних математичних операцій і обчислення складних функцій апаратним засобом за одну команду; використання нетрадиційних систем числення і засобів подання інформації; організація обчислень за принципом цифрової аналогії й ін.

Апаратні засоби як універсальних, так і спеціалізованих обчислювальних систем характеризуються ієрархічним принципом побудови, тобто наявністю сукупностей таких одиниць (елементів) кількості устаткування, що при об'єднанні їх у систему деякого рівня можуть розглядатися в якості елементів у системах більш високого рівня і т.д. Це, у свою чергу, дає можливість використовувати ієрархічний принцип опису структури і функціонування апаратних засобів. По цьому принципу, кожному рівню систем відповідає такий ступінь деталізації опису, що забезпечує точність опису до елементів системи даного рівня.

В обчислювальній техніці прийнято вважати *першим* ієрархічним рівнем, рівень електричних схем, де в якості елементів розглядаються електронні компоненти (транзистори, діоди, резистори та ін.). Засобом опису тут звичайно служить апарат теорії електричних і магнітних кіл.

На *другому* рівні – рівні логічних схем, де найменшими одиницями устаткування вважаються логічні елементи й елементи, що запам'ятовують.

Логічними елементами називають найпростіші комбінаційні схеми, функціонування яких описується одною перемикальною функцією (елементним

оператором). Елемент, що запам'ятовує – найпростіший пристрій пам'яті, що забезпечує запис, збереження і читання інформації, кількість якої дорівнює одній букві. Засобом опису на даному рівні є методи теорії перемикальних функцій і структурної теорії автоматів, що дозволяє деталізувати процеси опрацювання інформації до операцій над окремими буквами алфавітів.

Третій рівень – рівень операційних вузлів, що описується тими ж засобами, що і попередній рівень, із деталізацією інформаційних процесів до елементарних операцій (мікрооперацій) над окремими словами. У якості елементів на цьому рівні відіграють операційні вузли – апаратні засоби, що виконують одну або декілька елементарних операцій і побудовані з логічних і елементів, що запам'ятовують.

На *четвертому* рівні, який називається рівнем структурних схем, елементами вважаються операційні блоки, що об'єднують декілька операційних вузлів і виконують визначені закінчені дії, що вказуються командами програми. Склад і взаємодія апаратних засобів тут описуються з деталізацією до окремих операцій із набору команд обчислювальної системи, виконуваних як послідовності елементарних операцій. Засобом опису частіше усього служать найпростіші формальні мови типу операторних описів (у лінійно-рядковому або графічному варіантах).

П'ятий рівень – програмний, що припускає деталізацію процесів опрацювання інформації до команд з операційних ресурсів обчислювальної системи або до окремих програм. Елементами систем цього рівня є АЛП, ПК, пристрої пам'яті, периферійне, комунікаційне устаткування обчислювальної системи, а в якості засобів опису використовуються машинно-орієнтовані або процедурно-орієнтовані мови програмування.

Зауважимо, що логічні і запам'ятовуючі елементи, операційні вузли і блоки є структурно-функціональними одиницями устаткування обчислювальної системи. У конструктивному ж відношенні простішими одиницями апаратних засобів є модулі – частини електронного устаткування, що мають закінчене оформлення і стандартні засоби механічного й електричного сполучення з іншими подібними одиницями. Модульний принцип є основним принципом конструювання обчислювальних систем і інших пристроїв електронної техніки. Модульні конструкції можуть бути на всіх ієрархічних рівнях апаратних засобів, наприклад, модуль-процесор, модульний пристрій пам'яті, модуль АЛП, модуль-регістр, модуль логічних елементів (логічна мікросхема) і ін.

Лекція 2
СИСТЕМИ ЧИСЛЕННЯ У МАШИННІЙ АРИФМЕТИЦІ ЦИФРОВОЇ ОБЧИСЛЮВАЛЬНОЇ СИСТЕМИ

План

2.1. Поняття систем числення
2.2. Двійково-десяткові системи числення
2.3. Методи переведення чисел з однієї системи числення у іншу
2.4. Подання від'ємних чисел у
2.5. Форми подання чисел у обчислювальній системі

2.1. Поняття систем числення

Системою числення (численням, нумерацією) називають сукупність прийомів і правил для позначення і найменування чисел. У *будь-якій* системі числення число подають сукупністю символів, що називають цифрами. Кожній цифрі в записі числа *однозначно* співставляється визначена кількість, що задається цією цифрою. Цю кількість називають *кількісним* еквівалентом даної цифри.

Розрізняють *непозиційні* і *позиційні* системи числення. Систему числення називають *непозиційною*, якщо кожній цифрі в будь-якому місці запису числа однозначно відповідає той самий кількісний еквівалент. Такі системи є більш ранніми в історичному плані, наприклад, загальновідома римська нумерація. Проте непозиційні системи числення знаходять обмежене застосування у обчислювальній техніці, тому що вони характеризуються дуже складними і громіздкими алгоритмами подання чисел і виконання арифметичних операції.

Систему числення називають *позиційною*, якщо одній і тій же цифрі відповідають різноманітні кількісні еквіваленти в залежності від номера місця розташування (розряду) цієї цифри в записі числа.

У принципі можливі також *частково-позиційні* системи, у яких для однієї множини цифр кількісні еквіваленти сталі, а для іншої множини цифр вони залежать від їхнього місця розташування в записі числа.

Для визначення кількісного еквівалента повного запису числа використовується деяка функція від кількісних еквівалентів сукупності цифр у його записі. Якщо цією функцією є функція *додавання*, то систему називають *адитивною*, якщо ж функція *множення,* систему називають *мультиплікативною*. Для більшості існуючих систем числення зазначена функція є функцією *десяткового додавання*. У цьому випадку для перебування кількісного еквівалента числа необхідно просумувати усі кількісні еквіваленти цифр у його записі по правилах десяткової арифметики.

Якщо в позиційній системі числення кожна цифра має свій визначений символ, то її називають системою з безпосереднім поданням цифр. Поряд із цим існують системи з кодованим поданням цифр. У таких системах кожна цифра кодується визначеною комбінацією декількох символів, що, як правило, являють собою цифри іншої системи числення.

Найбільш поширеними у обчислювальній техніці є однорідні позиційні системи числення. В усіх розрядах числа, поданого в однорідній системі, вико-

ристовуються цифри з тої самої множини. Наприклад, у звичайній десятковій системі у всіх розрядах будь-якого числа використовуються цифри з множини {0, 1..., 9}, у двійковій системі – цифри з множини {0, 1} і т.п. У однорідній позиційній системі при безпосередньому поданні цифр число записується у вигляді:

$$X=x_S x_{S-1}\ldots x_1 x_0, x_{-1}\ldots x_{-m} \quad (4)$$

Кількісний еквівалент, що відображається цим записом, визначається так:

$$X=k^S x_S+ k^{S-1}x_{S-1}+ \ldots +k^1 x_1+ k^0 x_0+ k^{-1}x_{-1}+ k^{-m}x_{-m}= \sum_{i=-m}^{S} k^i x_i \quad (5)$$

де x_i - цифри *i*-го розряду запису числа, що приймають значення з визначеної множини, а *k* називається основою системи числення і дорівнює кількості цифр, що використовується у даній системі. Розмір k_i прийнято називати вагою *i-го* розряду. У *n-розрядному* слові (4), де $n = s + m + 1$, ціла частина числа подана s+1 розрядами зліва від коми, а дробова частина – *m* розрядами справа від коми. У даному випадку вага *i*-го розряду в *k* разів більше ваги *i*-1-го розряду. Таку систему числення називають системою з *природним порядком ваг* (*природною*). Існують також система зі *штучним порядком ваг*, для яких зазначене співвідношення ваг сусідніх розрядів не є обов'язковим.

До застосовуваного у ВТ системах числення пред'являють такі очевидні вимоги:

Однозначність. Кожному числу повинно відповідати єдине його подання в заданій системі і навпаки.

Скінченність. Кожному цілому числу повинно відповідати слово скінченної довжини.

Ефективність. Повинен існувати алгоритм, за допомогою якого за скінченне число кроків здійснювався б перехід від представлення числа скінченної довжини до самого числа. При переході від числа до його представлення повинні існувати алгоритми, які для цілого числа реалізують цей перехід за скінченне число кроків, а для дробового числа – за скінченне число кроків дозволяють одержати подання числа, кількісний еквівалент якого відрізняється від числа не більш, ніж на заданий розмір похибки.

Системи числення з натуральною основою, що задовольняють вимогам однозначності, скінченності й ефективності, називають *канонічними.* Для таких систем цифра 0 (нуль) є обов'язковою, а кількість різноманітних цифр дорівнює основі *k.*

Практика розробки й експлуатації обчислювальних систем показує, що обчислювальні машини, призначені для вирішення задач, у яких кількість обчислювальних операцій, що припадають на один символ введеної-виведеної інформації, велика, як правило, проектуються з використанням 2-ї системи числення. У обчислювальних системах, де час введення-виведення даних значно перевищує час опрацювання інформації, а також у малих універсальних , що

характеризуються інтенсивним обміном інформацією між людиною і машиною застосовується десяткова система числення.

2.2. Двійково – десяткові системи числення

Побудова десяткових , у яких цифри відображалися б сигналами, квантованими по десятьох рівнях, є практично доцільним тільки при наявності технічних засобів (схем) для подання чисел, основні техніко-економічні характеристики котрих (вартість, надійність, швидкодія й ін.) були б не гірші відповідних характеристик схем для подання 2-х цифр. Проте, як правило, деякі основні характеристики відомих схем для такого представлення десяткових цифр поступаються характеристикам 2-х схем. Тому при побудові десяткових обчислювальних систем використовують кодування 2-ми цифрами десяткових цифр.

Очевидно, що будь-який *k*-ічній цифрі можна поставити у відповідність будь-яке з 2^h двійкових чисел, що записуються за допомогою *h* розрядів. Отже, число способів двійкового кодування *k* цифр дорівнює числу розміщень із 2^h по *k*, тобто $\frac{2^h!}{(2^h-k)!}$. Це число дуже швидко росте зі збільшенням *k*, що унеможливлює аналіз ручними методами всіх способів двійкового кодування навіть для найбільш важливого випадку при $k = 10$.

Тому найбільшого поширення у обчислювальній техніці одержали двійково-десяткові системи числення, у яких десяткові цифри записуються як чотирирозрядні двійкові числа – двійкові тетради. Як показують теоретичні дослідження і досвід розробки десяткових обчислювальних систем, найкращі результати можуть бути отримані при використанні двійково-десяткових кодів (ДДК), що володіють властивостями *одиничності, упорядкованості, парності, додатковості і зваженості* (ці властивості називають також властивостями або умовами *Рутисхаузера*).

1. ДДК має властивість одиничності, якщо між десятковою цифрою і комбінацією двійкових цифр установлена взаємнооднозначна відповідність. Дана властивість забезпечує можливість роботи з ДДК і полегшує процес декодування.

2. Упорядкованість ДДК складається у виконанні однієї з умов

$0_{(2)}<1_{(2)}<\ldots<9_{(2)}$; $0_{(2)}>1_{(2)}>\ldots>9_{(2)}$

для двійкових представлень $0_{(2)}, 1_{(2)},\ldots 9_{(2)}$ десяткових цифр 0, 1, ..., 9. Наявність упорядкованості ДДК необхідна для реалізації логічних операцій.

3. Властивість парності ДДК повинна виявлятися в тому, щоб усім парним десятковим цифрам відповідали або тільки парні, або тільки непарні двійкові числа. Аналогічно для непарних. Ця властивість спрощує виконання операцій множення, розподілу й округлення.

4. Сутність властивості доповнення ДДК полягає в наступному: якщо сума двох десяткових цифр дорівнює 9, то перехід від двійкового подання однієї цифри до двійкового подання іншої повинно здійснюватися шляхом інвертування (тобто заміни 0 на 1 і навпаки) двійкових розрядів. Це полегшує формування оберненого і додаткового кодів, що використовується при операції алгебраїчного додавання.

5. ДДК називають зваженим, якщо кожному з h розрядів двійкового представлення $x_h x_{h-1}...x_1$ десяткової цифри X поставлені у відповідність ваги, a_h, a_{h-1}, $...a_1$, причому

$$X = a_h x_h + a_{h-1} x_{h-1} + \ldots + a_1 x_1 \qquad (6)$$

У сучасних обчислювальних системах, що опрацьовують не тільки числову інформацію, але і текстову (алфавітно-цифрову), окремі символи частіше усього подають словами довжиною в один байт. Тому один байт дорівнює двом двійковим *тетрадам*, а отже в обчислювальній системі із байтовим поданням інформації можна записувати дві десяткові цифри в один байт.

2.3. Методи переведення чисел з однієї системи числення у іншу

Розглянемо спочатку методи переведення чисел з системи числення з основою k_1 в систему з основою k_2. Через $X_{(k_1)}$ будемо позначати число X в k_1-ій системі.

Права частина виразу (5) визначає правило обчислення кількісного еквівалента числа, записаного у формі (4). На цьому заснований один із методів переведення чисел. Для переведення числа в систему з основою k_2 необхідно записати $X_{(k_1)}$ у формі (5); замінити цифри x_i і основу k_1 їхніми k_2-ми еквівалентами, а потім обчислити вираз (5) по правилах системи числення з основою k_2.

Приклад 1. перевести число $X_{(2)}$ = 1011,1001 у десяткову систему числення

$1011{,}1001 = 1\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 + 1\cdot2^{-1} + 0\cdot2^{-2} + 0\cdot2^{-3} + 1\cdot2^{-4} = 8 + 0 + 2 + 1 + 0{,}5 + 0 + 0 + 0{,}0625;$

$X_{(10)} = 11{,}5625.$

Описаний метод переведення чисел з однієї системи в другу одержав назву методу *безпосереднього заміщення*. Цей метод зручно використовувати у випадку, коли $k_1 < k_2$ і при переведенні чисел у таку систему, де просто виконуються операції додавання і множення (наприклад, із двійкової системи в десяткову).

Для спрощення обчислення при цьому можна скористатися таким виразом, отриманим із формули 4:

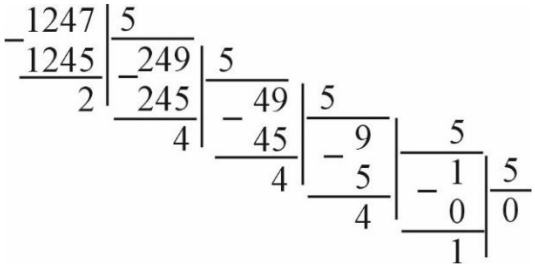
$$X = (\ldots((k x_n + x_{n-1})\,k + x_{n-2})\,k + \ldots + x_1)\,k + x_0 + (\ldots((k^{-1}x_{-m} + x_{-m+1})\,k^{-1} + x_{-m+2})\,k^{-1} + \ldots + x_{-1})\,k^{-1}$$

Проте при переведенні чисел у системи з «незвичними» основами, особливо у випадку $k_1 > k_2$, застосування цього методу пов'язане з досить громіздкими обчисленнями.

Метод переведення цілих чисел із системи з основою k_1 у систему з основою k_2 ($k_1 > k_2$) полягає в наступному. Число $X_{(k_1)}$, ділять на k_2 по правилах ділення з основою k_1 до одержання остачі. Якщо частка від ділення не нуль, то результат стає подільним і процес ділення на k_2 продовжують. Як тільки черго-

ва частка стане рівна нулю, процес ділення на k_2 припиняють. Остача, отримана при першому діленні на k_2, дає цифру розряду результату з вагою k залишок від другого ділення дає цифру результату з вагою k_2^0 і т.д. Остання остача є старшою цифрою результату.

Приклад 2. Перевести число $X_{(10)} = 1247$ у п'ятіркову систему числення.



$X_{(5)}$=14 442

2.4. Подання від'ємних чисел у обчислювальних системах

Від'ємне число можна подати двома різноманітними способами. Природно, наприклад, самий лівий, старший біт зробити *знаковим бітом* (sign bit), встановивши його значення рівним нулю для додатніх чисел і одиниці – для від'ємних, а інші біти відвести під абсолютний розмір числа. При цьому +45 буде подане як двійкове число 0000000000101101, а -45 – як 1000000000101101. Назвемо це «звичайним» шляхом подання двійкового числа з відомого.

Частіше для від'ємного числа використовується подання за допомогою *двійкового доповнення* (ones-complement) вже описаного звичайного представлення. Щоб одержати таке представлення від'ємного числа, потрібно спочатку знайти його «звичайне» представлення, потім обернути усі біти («доповнити» їх), рівні нулю, на одиницю, а всі рівні одиниці – на нуль, і, нарешті, добавити одиницю.

Для чого ж знадобилося подавати числа настільки вигадливим засобом? Насправді, для простоти.

У табл.1 приведені числа, від +5 до -6. Десяткове представлення числа дано в першому стовпчику, двійкове – в другому, і представлення в оберненому додатковому коді – у третьому.

І в першому, і в другому стовпчиках перший біт – знаковий, і одиниця в ньому позначає від'ємне значення.

Незвичайна в оберненому додатковому коді роль знакового біту. Зверніть увагу, що додавання одиниці до будь-якого числа третього стовпчика (рахуючи усі біти, включаючи знаковий, що належить звичайному 16-бітному додатному цілому), дає число в рядку над даними. Аналогічно, при вирахуванні одиниці утворюється число рядком нижче. І усе це справедливо і для додатних і для від'ємних чисел.

При спробі проробити ту ж операцію з числами в середньому стовпчику для одержання того ж результату доведеться вводити різноманітні правила для від'ємних і додатних чисел, а це істотно ускладнить виконання арифметичних дій із числами в «звичайному» представленні. Тому комп'ютери звичайно будуються таким чином, щоб час витрачався на представлення від'ємних чисел в

оберненому додатковому коді, оскільки пізніше це дасть значну економію при виконанні з ними арифметичних операцій.

Таблиця 1

Десяткові додатні і від'ємні числа подані в прямому і оберненому коді

Десяткове число	**«Очевидний» двійковий запис**	**Обернений додатковий код**
5	00000101	00000101
4	00000100	00000100
3	00000011	00000011
2	00000010	00000010
1	00000001	00000001
0	00000000	00000000
-1	10000001	11111111
-2	10000010	11111110
-3	10000011	11111101
-4	10000100	11111100
-5	10000101	11111011
-6	10000110	11111010

2.5. Форми подання чисел у обчислювальних системах

В даний час в обчислювальній техніці переважно використовуються дві форми подання чисел.

Перша з них одержала назву подання чисел із *фіксованою комою (природної форми)*, а друга – подання чисел із *плаваючою комою (нормальної або напівлогарифмічної форми).*

Будь-яке число X в позиційній системі числення з основою k можна записати як:

$$X = k^P M. \qquad (7)$$

де M називають мантисою числа X, а P – порядком того ж числа.

Якщо в обчислювальній системі кожному числу X однозначно відповідає мантиса M, а порядок фіксований для всіх чисел, то говорять, що число подане у формі з *фіксованої комою*.

Якщо ж кожному числу однозначно відповідає пара P і M, то таке представлення називають формою з *плаваючою комою*.

При представленні чисел у формі з фіксованою комою положення коми зберігається незмінним для всіх чисел, із якими оперує обчислювальна система. З метою спрощення обчислення масштабних коефіцієнтів кому звичайно фіксують перед старшим розрядом мантиси або після молодшого.

У першому випадку обчислювальна система оперує з числами, що менше одиниці. Якщо число розрядів для запису мантис складає n, то мінімальне (по абсолютному розміру) число, що може бути подане в машині, дорівнює k^{-n}, а максимальне $-1-k^{-n}$. Слід зазначити, що при такому представленні чисел зручно

виконувати операцію множення, тому що при цьому не відбувається переповнення розрядної сітки.

В другому випадку в машині виконуються операції над цілими числами, абсолютні розміри яких знаходяться в межах від 0 до k^{-n}-1. Іноді кому зрушують за межі розрядної сітки вправо на l розрядів, тобто вводять l умовних розрядів. При цьому абсолютний розмір максимального числа в машині складає $k^{n+l} - k^l$, а мінімального $-k^l$.

Розрядна сітка для представлення чисел (формат даних) в обчислювальній системі із фіксованою комою складається з двох частин: один розряд для представлення знаку, інші розряди – для представлення мантиси M.

При поданні чисел у формі з плаваючою комою порядок P може бути позитивним або негативним цілим числом. Мантиса ж у більшості випадків є позитивним або негативним правильним дробом, причому $k^{-1} \leq M < 1$. Якщо в обчислювальній системі для запису порядку використовуються m розрядів, а для запису мантиси – n розрядів, то в цій машині може бути подане таке максимальне (по абсолютному розмірі) число:

$$k^{k^m-1}(1-k^{-n}) = k^{k^m-1} - k^{k^m-n-1} \qquad (8)$$

У свою чергу, мінімальне (по абсолютному розмірі) число, що відрізняється від нуля, у такій машині дорівнює $k^{-k^m+1}k^{-1} = k^{-k^m}$.

Крім зазначених вище розрядів для представлення порядку і мантиси в розрядній сітці обчислювальної системи із плаваючою комою є також два розряди для представлення знаків порядку і мантиси.

У обчислювальних системах, де використовується представлення чисел у формі з плаваючою комою, арифметичні операції виконуються як над мантисами чисел, так і над їх порядками. Часто також необхідно виконувати операцію нормалізації чисел, сутність якої складається у виконанні умови $k^{-1} \leq M < 1$. Тому арифметично-логічні пристрої обчислювальної системи із плаваючою комою є більш складними і менше швидкодіючими, ніж обчислювальні системи із фіксованою комою. З іншого боку, в обчислювальній системі із фіксованою комою через масштабування виникають труднощі при підготовці задач до розв'язку.

Для двох розглянутих форм подання чисел властиві свої позитивні якості і хиби, тому в сучасних універсальних великих обчислювальних системах використовують обидві форми.

Числа з плаваючою комою в обчислювальних системах ЄС можуть бути двох форматів: короткого і довгого.

Короткий формат має 32 розряди, із яких 7 розрядів призначені для подання порядку, 24 – для подання мантиси і 1 розряд – для знаку мантиси. Мантиси негативних чисел при цьому рекомендуються прямим кодом.

Довгий формат, використовується для обчислень із підвищеною точністю, має 64 розряди. Для порядку приділяється стільки ж розрядів, що і при короткому форматі, а для мантиси – 56 розрядів. Кома знаходиться зліва від старшого розряду мантиси.

Лекція 3
ОПЕРАЦІЙНІ ВУЗЛИ КОМБІНАЦІЙНОГО ТА ПОСЛІДОВНІСНОГО ТИПУ

План

3.1. Технічні засоби цифрових
3.2. Пристрої комбінаційного типу
3.3. Тригери
3.4. Регістри
3.5. Лічильники
3.6. Суматори

3.1. Технічні засоби цифрових обчислювальних систем

Технічні засоби цифрових обчислювальних систем у залежності від виду функцій, що описують їхню роботу, прийнято розділяти на два класи: *пристрої комбінаційного типу* (комбінаційні, логічні, функціональні схеми, схеми без пам'яті) *і послідовнісного типу* (цифрові автомати, схеми з пам'яттю).

У *пристроях комбінаційного типу* вихідні сигнали залежать тільки від вхідних сигналів у даний момент дискретного часу і не залежать від вхідних сигналів, що діяли в попередні моменти часу.

У *пристроях послідовнісного типу* вихідні сигнали визначаються як вхідними сигналами, так і станом автомата, що, у свою чергу, залежить від вхідних сигналів, що діяли раніше.

Математичний апарат, що забезпечує розв'язок задачі проектування цифрових пристроїв на логічному й операційному рівнях, розробляється в теорії перемикальних функцій і цифрових автоматів.

3.2. Пристрої комбінаційного типу

Функціонування цифрових обчислювальних пристроїв комбінаційного типу, що мають n входів і m виходів, у загальному випадку може бути описано системою функцій виду

$$Y_i=f_i(x_1,x_2, \ldots, x_n) \qquad (9)$$

де значення функцій Y_i визначають значення вихідних сигналів, а набори аргументів ($x_1, x_2, \ldots, x_n$) відповідають вхідним сигналам. Як значення функції Y_i, так і аргументи x_i можуть приймати тільки кінцеве число значень (*зазвичай* 0 і 1). Такі функції одержали назву *перемикальних (нульових, двозначних, логічних)*.

Перемикальні функції частіше усього задають за допомогою таблиць, які називаються *таблицями істинності*, шляхом перерахування їхніх значень на всіх наборах значень аргументів. Число аргументів однозначно визначає число різноманітних функцій від цих аргументів. При числі аргументів, рівному n, число їхніх різноманітних з'єднань складе $2n$, а число функцій уже $4n$.

Всі можливі логічні функції для двох змінних перераховані в табл. 2

Таблиця 2

Всі можливі логічні функції для двох змінних

Таблиця істинності					**Позначення логічної операції**	**Назва функції (операції)**
X1	0	0	1	1		
X2	0	1	0	1		
Y0	0	0	0	0	—	Константа 0
Y1	0	0	0	1	$X1 \cdot X2$, $X1 \wedge X2$, X1&X2	Логічне множення, логічне І, кон'юнкція
Y2	0	0	1	0	$X1 \cdot \overline{X2}$	Заборона по X2
Y3	0	0	1	1	X1	Тотожність, змінна X1
Y4	0	1	0	0	$X2 \cdot \overline{X1}$	Заборона по X1
Y5	0	1	0	1	X2	Тотожність, змінна X2
Y6	0	1	1	0	$X1 \oplus X2$, X1+X2	Сума по модулю 2, нерівнозначність
Y7	0	1	1	1	$X1 \vee X2$	Логічна сума, логічне АБО; диз'юнкція
Y8	1	0	0	0	$\overline{X1 \vee X2}$	Логічне АБО — НЕ;
Y9	1	0	0	1	$X1 \equiv X2$	Еквівалентність, рівнозначність
Y10	1	0	1	0	$\overline{X2}$	Логічне НЕ; логічне заперечення; інверсія X2
Y11	1	0	1	1	$X2 \vee \overline{X1}$	Імплікація по X1
Y12	1	1	0	0	$\overline{X1}$	Логічне НЕ; логічне заперечення, інверсія X1
Y13	1	1	0	1	$X2 \vee \overline{X1}$	Імплікація
Y14	1	1	1	0	$\overline{X1 \cdot X2}$	Логічне І — НЕ; функція Шеффера
Y15	1	1	1	1	—	Константа 1

З усіх функцій, наведених у табл. 2, найбільш цікаві функції І-НЕ або АБО-НЕ, тому що кожна з них утворює функціонально повну систему. При проектуванні комбінаційних обчислювальних пристроїв із використанням мікросхем малого ступеня інтеграції, найбільш важливою, з інженерної точки зору, є задача синтезу комбінаційних схем із *n* входами й одним виходом із мікросхем, що реалізують елементні оператори І, АБО, НЕ і їхньої комбінації І-НЕ, АБО-НЕ і т.д.

При цьому *комбінаційною схемою* називають сукупність логічних елементів, що реалізують задану систему перемикальних функцій і з'єднаних по правилах, що відповідають суперпозиції функцій.

Логічний елемент – це електронний пристрій, що реалізує одну з логічних операцій.

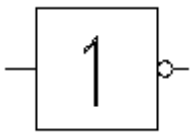
Будь-який цифровий пристрій може бути побудований з логічних елементів одного типу, на базі елементів І-НЕ або АБО - НЕ, на практиці не часто обмежуються одним типом елементів.

На принциповій схемі логічний елемент зображують прямокутником, усередині якого знаходиться зображення покажчика функції. Лінії з лівої сторони прямокутника показують входи, із правої – виходи елемента. Якщо якийсь вхід позначений колом, то це значить, що функція виконується для сигналу низького рівня (негативна логіка). Якщо колом позначений вихід, то елемент робить логічне заперечення (інверсію) результату операції, зазначеної усередині прямокутника.

Основні логічні елементи, що використовуються в цифрових пристроях:

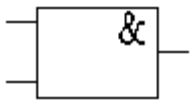
1). Елемент НЕ (NOT Gate): $Y=\overline{X}$;

X	**Y**
0	1
1	0



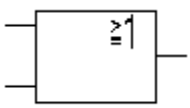
2). Елемент І (AND Gate): $Y=X1\cdot X2$;

X1	**X2**	**Y**
0	0	0
0	1	0
1	0	0
1	1	1



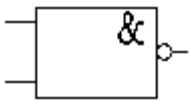
3). Елемент АБО (OR Gate) $Y=X1\vee X2$

X1	**X2**	**Y**
0	0	0
0	1	1
1	0	1
1	1	1



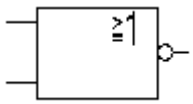
4). Елемент І - НЕ (NAND Gate): $Y=\overline{X1\cdot X2}$

X1	**X2**	**Y**
0	0	1
0	1	1
1	0	1
1	1	0



5). Елемент АБО-НЕ (NOR Gate): $Y=X1 \vee X2$

X1	X2	Y
0	0	1
0	1	0
1	0	0
1	1	0



Базуючись на законах алгебри логіки можна показати, що будь-який із вище перерахованих елементів можна зібрати з базових двовхідних елементів І-НЕ.

3.3. Тригери

Основним елементом пристроїв послідовнісного типу, де значення функції залежить не тільки від значення змінних в даний момент часу (даний такт), але і від їхньої послідовності в попередні моменти (такти), є тригерний елемент пам'яті, або просто тригер.

Тригер – це пристрій із двома стійкими станами, призначений для запису і збереження інформації. Під дією вхідних сигналів тригер може переключатися з одного стійкого стану в інший. При цьому напруга на його виході стрибкоподібно змінюється.

Як правило, тригер має два виходи, прямий Q і інверсний $\overline{Q}$. Число входів залежить від виконуваних функцій.

За способом запису інформації тригери поділяють на *асинхронні* і *синхронні.*

У асинхронних тригерах інформація може змінюватися в будь-який момент часу при зміні вхідних сигналів.

У синхронних тригерах інформація на виході може змінюватися тільки у визначені моменти часу, що задаються синхронізуючим сигналом.

Існує велике число різноманітних тригерів із різноманітними функціональними можливостями, проте в основі всіх схем лежить основний (базовий) асинхронний RS-тригер.

Асинхронний RS-тригер може бути побудований на двох логічних елементах АБО-НЕ або І-НЕ (рис. 4).

Тригер має два входи: *S* – вхід установки в одиничний стан (від англ. set - установка) і *R* – вхід скидання в нульовий стан (від англ. reset-скидання).

З схеми очевидно, що при *S*=*R*=1 обидва вихідних сигнали рівні нулю, що не дозволяє однозначно визначити стан системи. Тому комбінація вхідних сигналів *S*=*R*=1 є забороненою.

Опис роботи RS-тригерів можна зробити і за допомогою таблиці перемикання.

Широкому використанню асинхронного RS-тригера в якості самостійного пристрою заважають властиві йому серйозні недоліки: наявність забороненої комбінації вхідних сигналів, подача інформації з двох окремих ланцюгах (*R, S*).

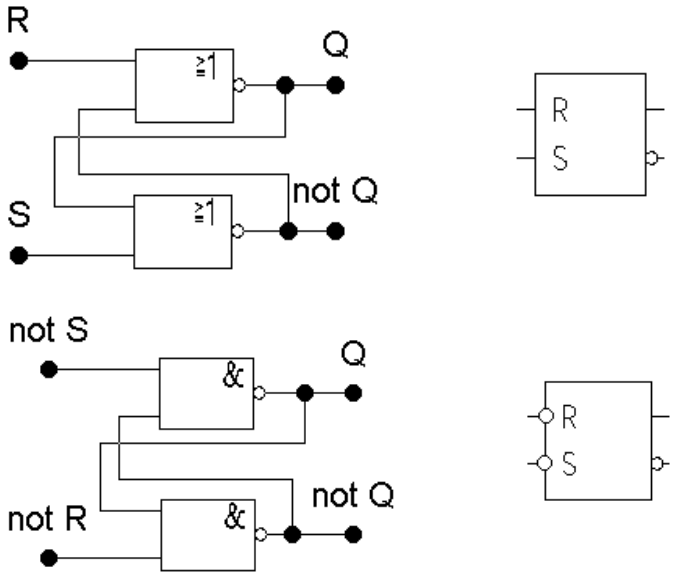


Рис. 4. Схеми асинхронного RS-тригера

Таблиця 3

Стани асинхронного RS-тригера

Вхід		Вихід		Режим роботи
АБО - НЕ		Q	$\overline{Q}$	
S	R			
0	0	0	0	Збереження
1	0	1	0	Запис 1
0	1	0	1	Запис 0
1	1	X	X	Заборонений $Q = \overline{Q}$

Синхронний D-тригер вільний від недоліків *RS*-тригера. *D*-тригер утворений із *RS* -тригера і вхідної комбінаційної схеми на двох логічних елементах І-НЕ. Сигнали, призначені для занесення в тригер, надходять на інформаційний вхід *D.* На вхід синхронізації *C* подають синхроімпульси, що визначають момент запису інформації.

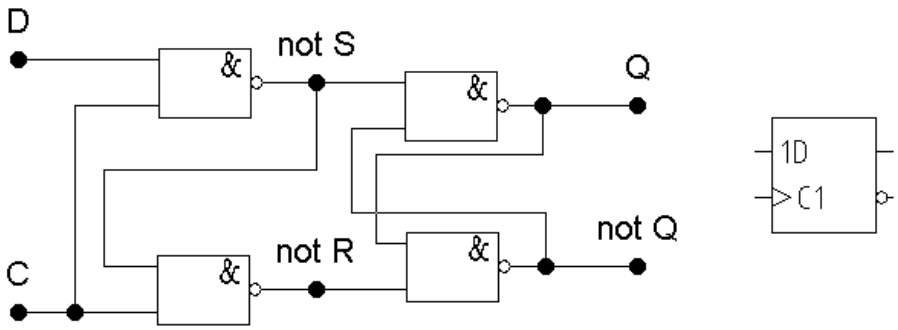


Рис. 5. Схема статичного *D*-тригера

Опис роботи тригера при різноманітних комбінаціях вхідних сигналів подано в табл. 4

Таблиця 4

Стани роботи *D*-тригера

C	D	Q	Режим роботи
0	0	Попереднє значення	Збереження
0	1	Попереднє значення	Збереження
1	0	0	Запис 0
1	1	1	Запис 1

З табл. 4 очевидно, що *D*-тригер знаходиться в режимі збереження при *C*=0 і в режимі запису при *C*=1. Такий тригер затримує вихідний сигнал до закінчення того такту, у який він був записаний. Звідси відбулася і назва *D*-тригера (від англ. delay-затримка). Якщо сигнал на вході зміниться під час дії синхроімпульсу, то в тригері виявиться записаною та інформація, що передувала закінченню синхроімпульсу. Завдяки цій властивості розглянутий тригер називається *статичним синхронним D-тригером.* Для нормальної роботи статичного *D-тригера* необхідно, щоб зміна інформації на *D*-вході відбувалася тільки при *C*=0.

Динамічний синхронний D-тригер виключає наскрізну передачу сигналу з *D*-входу на вихід тригера під час дії синхроімпульсу. У тригері з динамічним керуванням, інформація записується тільки в момент перепаду напруги на вході синхронізації.

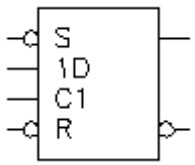


Рис. 6. Умовне позначення динамічного *D*-тригера

Опрацювання цифрової інформації в складних системах відбувається у вигляді послідовного виконання окремої елементарних операції. Ці елементарні операції виконуються операційними елементами. Операційні елементи, або вузли, цифрових пристроїв утворені з логічних елементів комбінаційної і послідовнісної логіки

Основний набір елементарних операцій невеликий:

Установка – запис в операційний елемент двійкового коду якоїсь константи.

Приклад – установка нуля в усіх розрядах лічильника.

Передача - прийом – перезапис коду числа з одного операційного елемента в інший.

Зсув – зміна положення розрядів коду щодо початкового.

Рахунок – збільшення або зменшення коду числа на виході операційного елемента при надходженні на його вхід імпульсної послідовності.

Перетворення – переклад коду числа з однієї системи кодування в іншу.

Розподіл – адресна передача сигналів від багатьох джерел одному споживачу або від одного джерела кільком споживачам.

Додавання – знаходження суми двох чисел, поданих у двійковому коді.

Вузли, що виконують основні елементарні операції, також називаються основними вузлами цифрових пристроїв. До них відносяться *регістри, лічильники, перетворювачі кодів, мультиплексори і суматори.*

3.4. Регістри

Регістром називається впорядкована послідовність тригерів, призначена для збереження слів і виконання мікрооперацій над ними.

Мікрооперація – це елементарна машинна дія, в результаті якої змінюється значення слова, або відбувається його пересилка.

Загальна структура регістра зображена на рис. 7.

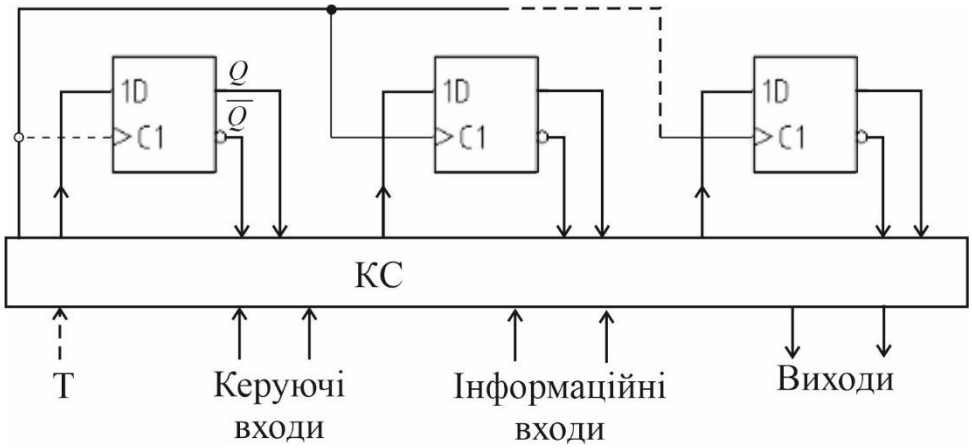


Рис. 7. Загальна структура регістра

Кількість розрядів (тригерів) називають довжиною регістра. Регістр може знаходитись в 2^n станах, тобто в нього можна записати 2^n слів.

Розрізняють синхронні і асинхронні регістри. В синхронних регістрах мікрооперації виконуються по тактуючому сигналу *T*.

В асинхронних тригерах такий тактуючий сигнал *T* відсутній. Кожна мікрооперація виконується під дією власного тактуючого сигналу. Кількість таких сигналів рівна кількості мікрооперацій.

При виконанні мікрооперацій в кожному розряді регістра відбувається однакове перетворення інформації.

Найбільш часто на регістрах виконують мікрооперації занесення (прийому, запису) слова паралельним кодом, зсуву слова, а також встановлення початкового (як правило нульового) стану.

Занесення слова відбувається через інформаційні входи D_i, (i = 1, n).

Регістри, на яких виконується операція зсуву, називаються *зсувними*. Зсув слова може бути проведений вліво або вправо на певну кількість розрядів одночасно. Регістри, які мають ланцюги як лівого так і правого зсуву, називаються *реверсивними.*

Виходами регістра бувають виходи тригерів, але в багатьох випадках КС включає елементи, які здійснюють видачу інформації. Видача слова може бути здійснена в прямому або оберненому коді. Для видачі слова в прямому коді, до виходів регістра підключають прямі виходи тригерів, а в оберненому коді – ін-

версні виходи.

3.5. Лічильники

Лічильником називають послідовну схему, призначену для виконання мікрооперацій рахунку і збереження слів. Кількість дозволених станів лічильника називають його періодом, модулем, або коефіцієнтом перерахунку *K*.

Лічильники можуть бути побудовані на основі тригерів із спеціальними міжрозрядними зв'язками.

Основними часовими характеристиками лічильників є: f – максимальна частота поступання рахуючих сигналів; t – час переходу лічильника з одного стану в інший.

Лічильники з спеціальними міжрозрядними зв'язками класифікуються за різними ознаками.

По характеру мікрооперації рахунку лічильники поділяються на *сумуючі, від'ємні та реверсивні.*

При надходженні наступного сигналу *X* місткість сумуючого лічильника збільшується на 1, а від'ємного зменшується на 1. Реверсивний лічильник може виконувати як мікрооперацію додавання так і мікрооперацію віднімання, в залежності від значення сигналу на керуючому вході *Y* (наприклад, при $Y = 1$ виконується додавання, а при $Y = 0$ – віднімання).

В залежності від основи системи числення, в якій відбувається мікрооперація рахунку, розрізняють двійкові лічильники, двійково-десяткові і т.д.

Лічильники класифікуються по схемних ознаках. Для побудови лічильників використовуються синхронні тригери з внутрішньою затримкою, що дозволяє використовувати на один розряд двійкового лічильника один тригер.

За способом організації ланцюгів переносу між розрядами лічильники поділяються на такі типи:

- з послідовним переносом;
- з наскрізним переносом;
- з паралельним переносом;
- з груповим переносом.

В лічильниках з послідовним переносом перенесення в старший розряд формується тільки після перемикання тригера в попередньому розряді. Тобто тригери перемикаються не одночасно. При проектуванні таких лічильників виникають труднощі, пов'язані з необхідністю аналізу не тільки логічного рівня сигналів, які формуються в схемі, але й моментів зміни рівнів сигналів.

На рис.8 зображена функціональна схема *n*-розрядного сумуючого лічильника з послідовним переносом, побудованого на синхронних *T*-тригерах, які перемикаються по від'ємному перепаду тактуючого сигналу.

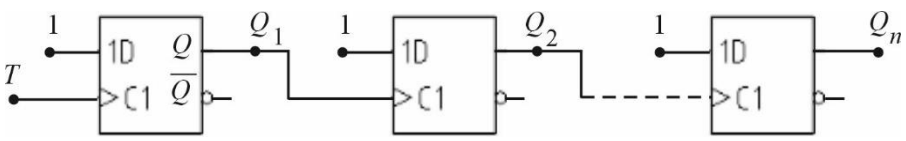


Рис. 8. Схема *n*-розрядного сумуючого лічильника з послідовним переносом, побудованого на синхронних *T*-тригерах.

В лічильниках з паралельним переносом аргументами функцій переносів для кожного розряду є тільки сигнали на виходах тригерів відповідних розрядів. Переноси для всіх розрядів лічильника формуються одночасно (при умові, що всі логічні елементи в схемі мають однаковий час перемикання).

Ланцюги наскрізного переносу організовуються таким чином, щоб функція переносу *i*-го розряду лічильника була аргументом функції переносу (*i*+1)-го розряду. В такому випадку сигнали переносів для кожного розряду формуються по черзі. Лічильники з наскрізним переносом потребують меншої кількості входів логічних елементів для організації ланцюгів переносу, але поступаються лічильникам з паралельним переносом в швидкодії.

3.6. Суматори

Суматор – це операційний вузол, який виконує мікрооперацію арифметичного додавання двох чисел (слів). Додавання *n* -розрядних чисел зводиться до виконання порозрядних операцій:

$$\left.\begin{matrix} s_i = x_i + y_i + z_i \\ p_i = 0 \end{matrix}\right\} x_i + y_i + z_i < k \qquad (10)$$

$$\left.\begin{matrix} s_i = x_i + y_i + z_i \\ p_i = 1 \end{matrix}\right\} x_i + y_i + z_i \geq k \qquad (11)$$

де s_i – значення суми в i -му розряді; z_i – перенесення з молодшого розряду; p_i – перенесення в старший розряд; k – основа системи числення; x_i, y_i – порозрядні значення складових числа.

В залежності від системи числення розрізняють двійкові, трійкові, десяткові, двійково-десяткові та інші суматори. Також суматори розрізняють за способом організації переносу додавання однорозрядних складових, за способом обробки багаторозрядних складників.

Лекція 4
АРХІТЕКТУРА МІКРОПРОЦЕСОРА

План

4.1. Структурна схема мікропроцесора.
4.2. Регістри мікропроцесора.
4.3. Акумулятор.
4.4. Лічильник команд.
4.5. Регістр адреси пам'яті.
4.6. Регістр команд.
4.7. Регістр стану.
4.8. Буферні регістри АЛП.
4.9. Регістри загального призначення.
4.10. Внутрішня шина даних мікропроцесора.

4.1. Структурна схема мікропроцесора

Структурна схема мікропроцесора дає можливість наочно роздивитися його роботу з виконання двох основних функцій: обробки і маніпулювання даними.

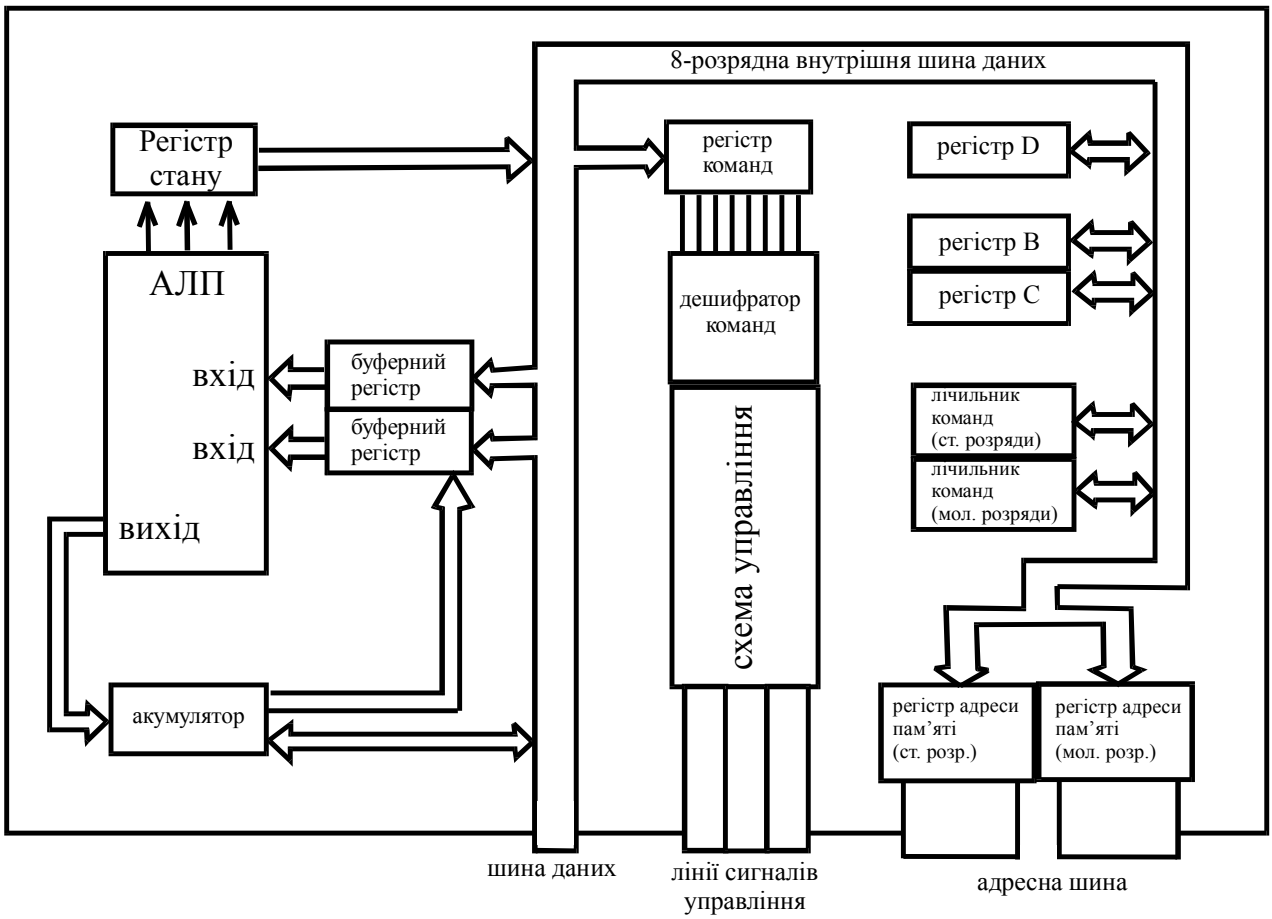


Рис. 9. Структурна схема мікропроцесора

Мікропроцесор складається з трьох основних блоків: АЛП, декількох регістрів і пристрою керування. Для передачі даних між цими блоками мікропроцесора використовується внутрішня шина даних.

Арифметико-логічний пристрій (АЛП) виконує одну з головних функцій мікропроцесора – опрацювання даних. Відповідно до рис. 9, АЛП має два вхідних порти, позначених як «вхід», і один вихідний порт – «вихід». Призначення вхідного порту – введення слова даних в АЛП, а вихідного порту – виведення такого слова. Подібні схеми мають один або декілька вхідних портів і єдиний вихідний порт. Обидва вхідних порти оснащені буферами, роль яких виконують регістри тимчасового збереження даних (буферні регістри). Кожен порт сполучений із своїм буферним регістром, здатним зберегти для АЛП одне слово даних.

Два вхідних порти дозволяють АЛП приймати дані або з внутрішньої шини даних мікропроцесора, або зі спеціального регістра, який називається акумулятором. Єдиний вихідний порт АЛП надає останньому можливість пересилати слово даних в акумулятор.

Акумулятор призначений для збереження слова даних, посланого в нього з вихідного порту АЛП або з пам’яті.

АЛП оперує одним або двома словами в залежності від виду виконуваної операції; відповідно він використовує і вхідні порти.

Перелік функцій АЛП залежить від типу мікропроцесора і різний для машин різних типів. Деякі АЛП спроможні виконувати багато різноманітних операцій, в інших набір операцій обмежений невеличким числом. Функції АЛП визначають архітектуру мікропроцесора.

Типовими операціями, виконуваними АЛП більшості мікропроцесорів, є:

додавання, віднімання, і, або, виключає або, інверсія, зсув вправо, зсув вліво, збільшення позитивне, збільшення негативне.

4.2. Регістри мікропроцесора

Регістри мікропроцесора беруть участь у реалізації основних логічних функцій мікропроцесора незалежно від їх кількості. Обмежимося розглядом шести основних регістрів.

Кожний регістр мікропроцесора може використовуватися для тимчасового збереження одного слова даних. Деякі регістри мають спеціальне призначення, інші – багатоцільове. Регістри останнього типу називаються регістрами *загального призначення* і можуть використовуватися по різному.

Кількість і призначення регістрів у мікропроцесорі залежить від архітектури останнього. Проте майже всі мікропроцесори мають ***шість основних регістрів***:

- *регістри стану,*
- *буферні регістри,*
- *регістри команд,*
- *регістри адреси пам’яті,*
- *лічильник команд,*
- *акумулятор*.

Інші регістри призначені для спрощення і полегшення роботи програміста.

4.3. Акумулятор

Акумулятор – головний регістр мікропроцесора, що використовується при різноманітних маніпуляціях із даними. Більшість арифметичних і логічних операцій здійснюється шляхом використання АЛП і акумулятора. Будь-яка з таких операцій над двома словами даних припускає розміщення одного з них в акумуляторі, а іншого в пам'яті або ще іншому регістрі.

Операцією іншого типу, що використовує акумулятор, є програмована передача даних з однієї частини мікропроцесора в іншу. Виконання такої операції здійснюється в два етапи: спочатку виконується пересилка даних із джерела в акумулятор, а потім з акумулятора – у пункт призначення.

Вище було показано, що мікропроцесор дозволяє використовувати АЛП для об'єднання даних в акумуляторі з іншими даними. Проте мікропроцесор може виконувати деякі дії над даними безпосередньо в акумуляторі. Наприклад, акумулятор може бути очищений шляхом запису двійкових нулів в усі його розряди, встановлений в одиничний стан за допомогою запису двійкових одиниць у всі розряди. Вміст акумулятора можна зміщатися вліво або вправо, одержувати його інвертоване значення, а та інше.

Акумулятор є найбільш універсальним регістром мікропроцесора. Дані надходять у нього з внутрішньої шини даних мікропроцесора. У свою чергу, акумулятор може посилати дані на цю шину.

Кількість розрядів акумулятора відповідає довжині слова мікропроцесора. Проте деякі мікропроцесори мають акумулятори подвійної довжини. Такий акумулятор можна розглядати або як один пристрій, або як два окремих акумулятори.

У деяких мікропроцесорів є група акумуляторів.

Перевага «багатоакумуляторних» мікропроцесорів у порівнянні з «одноакумуляторними» у тому, що перші надають можливість виконання операцій із передачею даних від акумулятора до акумулятора.

4.4. Лічильник команд

Лічильник команд стежить за тим, яка команда виконується, а яка підлягає виконанню.

Часто лічильник команд має набагато більше розрядів, ніж довжина слова даних мікропроцесора.

Лічильник команд сполучений із внутрішньою шиною даних мікропроцесора. Теоретично цей лічильник може одержувати дані про адреси програми з будь-якого блока мікропроцесора, під'єднаного до внутрішньої шини. Проте на практиці дані звичайно надходять у лічильник команд із пам'яті мікрообчислювальній системі.

На відміну від акумулятора, лічильник команд не може виконувати операції різноманітного типу. Набір його команд, що використовують, вкрай обмежені у порівнянні з подібним набором для акумулятора.

Після отримання команди з пам'яті мікропроцесор автоматично збільшує вміст лічильника команд. Це збільшення лічильник команд одержує саме в той

момент, коли мікропроцесор починає виконувати команду, тільки що отриману з пам'яті. Отже, починаючи з цього моменту, лічильник команд «вказує», якою буде наступна команда.

Лічильник команд може бути завантажений іншим вмістом при виконанні особливої групи команд. Може виникнути необхідність виконати частину програми, що «випадає» із послідовності команд основної, або головної, програми.

4.5. Регістр адреси пам'яті

При кожному звертанні до пам'яті мікро-обчислювальної системи *регістр адреси* пам'яті вказує адреси області пам'яті, що підлягають використанню мікропроцесором. Регістр адреси пам'яті містить двійкове число – адресу області пам'яті. Вихід цього регістра називається адресною шиною і використовується для вибору області пам'яті або в деяких випадках для вибору порту введення-виведення.

Протягом підциклу вибірки команди з пам'яті регістри адреси пам'яті і лічильника команд мають однаковий уміст, тобто регістр адреси пам'яті вказує місце розташування команди, що отримується з пам'яті. Після декодування команди лічильник команд одержує збільшення. Що ж стосується регістра адреси пам'яті, то він збільшення не одержує.

Для деяких команд адресація до пам'яті не потрібна. Така, наприклад, команда очищення акумулятора. При опрацюванні таких команд регістр адреси пам'яті використовується лише один разів – протягом підциклу вибірки команди з пам'яті.

У більшості мікропроцесорів регістри адреси пам'яті і лічильника команд мають однакове число розрядів. Такий регістр можна розділити на два окремих регістри, кожний із який має незалежне підключення до шини даних мікропроцесора. Один із цих регістрів називають регістром *старшого* байта (СБ), інший – регістром *молодшого* байта (МБ).

Оскільки регістр адреси пам'яті залучений до внутрішньої шини даних мікропроцесора, він може завантажуватися від різноманітних джерел. Більшість мікропроцесорів мають у своєму розпорядженні команди, що дозволяють завантажувати цей регістр умістом лічильника команд, регістра загального призначення або якоїсь області пам'яті. Деякі команди надають можливість змінювати вміст регістра адреси пам'яті шляхом виконання обчислень: нове значення вмісту цього регістра утворюється шляхом додавання або вирахування вмісту лічильника команд із числом, зазначеним у самій команді. Адресація такого типу одержала назву *адресації з використанням зсуву*.

4.6. Регістр команд

Регістр команд призначений винятково для збереження поточної виконуваної команди, причому ця функція реалізується мікропроцесором автоматично з початком циклу вибірка-виконання, що називається також машинним циклом.

Машинний цикл складається з двох підциклів – *вибірки і виконання*. Послідовність реалізації циклу вибірка-виконання така: спочатку команда отримується з пам'яті, потім лічильник команд настроюється на команду, що підлягає

виконанню. При отриманні команди з відповідної області пам'яті копія команди поміщається на внутрішню шину даних і пересилається в регістр команд. Після цього починається підцикл виконання команди, протягом якого дешифратор команд «читає» уміст регістра команд, повідомляючи мікропроцесору, що робити для реалізації команди.

Число розрядів регістра команд залежить від типу мікропроцесора: іноді воно збігається з числом розрядів слова даних, в інших випадках дорівнює лише 3 або 4.

4.7. Регістр стану

Наявністю ***регістра стану*** обчислювальна машина відрізняється від простого калькулятора. Зазначений регістр призначений для збереження результатів деяких перевірок, здійснюваних у процесі виконання програми. Розряди регістра стану приймають те або інше значення при виконанні операцій, що використовують АЛП і деякі регістри.

Запам'ятовування результатів згаданих перевірок дозволяє використовувати програми, що містять порушення природної послідовності виконання команд (переходи).

Це призвело до появи нового набору команд мікропроцесора. Ці команди призначені для зміни ходу виконання програми. Традиційний засіб використання цих спеціальних команд припускає завантаження лічильника команд новим вмістом.

Типовим прикладом операцій АЛП є арифметичні, при реалізації котрих можливе генерування одиничного біта переносу, формування нульового результату (двійкових нулів у всіх розрядах) або і те й інше одночасно.

4.8. Буферні регістри АЛП

Буферні регістри АЛП призначені для тимчасового збереження одного слова даних. Один із цих регістрів називається *буфером акумулятора* АЛП. Що стосується іншого буферного регістра, то в нього на тимчасове збереження надходять дані з внутрішньої шини мікропроцесора. Необхідність у такому регістрі викликана відсутністю в АЛП свого устрою, що запам'ятовує.

Якщо на вхід описаного вище буферного регістра можуть надходити дані тільки з внутрішньої шини даних мікропроцесора, то на вхід іншого буферного регістра, іменованого *буфером акумулятора*, дані можуть надходити, крім того, і з виходу акумулятора. Коли в арифметичній або логічній операції АЛП беруть участь два слова, одне з них надходить з акумулятора.

4.9. Регістри загального призначення.

Всі мікропроцесори мають шість типів описаних вище основних регістрів. На додаток до них деякі мікропроцесори мають у своєму розпорядженні інші регістри. Ці регістри одержали назву регістрів *загального призначення*. У деяких мікропроцесорах вони служать у якості пристроїв, що запам'ятовують, в інших функціональні можливості цих регістрів не поступаються можливостям акумулятора. У нашому випадку мікропроцесор має три регістри загального призначення: B, C і D.

Вибір конкретного регістра для виконання певного виду робіт визначається лише тим, який із них доступний і здається найбільш зручним.

Роль ***схем керування*** в мікропроцесорі полягає в підтримці необхідної послідовності функціонування всіх інших його ланок. По «розпорядженню» схем керування чергова команда витягається з регістра команд, визначається, що необхідно робити з даними, а потім генерується послідовність дій по виконанню поставленої задачі.

Звичайно робота схем керування мікро-програмується. Можна сказати, що схеми керування – це маленький мікропроцесор усередині мікропроцесора. Одна з головних функцій схем керування – декодування команди, що знаходиться в регістрі команд, за допомогою дешифратора команд.

Однією з важливих вхідних ліній керування, що з'єднують мікропроцесор із зовнішніми пристроями, є лінія зв'язку з генератором тактових імпульсів (таймером), що синхронізують у часі роботу мікропроцесора.

Крім зазначених вище дій, схеми керування виконують деякі інші спеціальні функції, такі, як керування послідовністю вмикання живлення, керування процесами переривань та ін..

4.10 Внутрішня шина даних мікропроцесора.

Структурна схема мікропроцесора показує, що внутрішня шина даних з'єднує між собою АЛП і регістри, здійснюючи передачу даних усередині мікропроцесора.

Кожен функціональний блок мікропроцесора завжди залучений до внутрішньої шини даних, проте скористатися нею можна тільки після одержання відповідного сигналу від схем керування. Майже усі функціональні вузли мікропроцесора мають двосторонній зв'язок із внутрішньою шиною даних, тобто вони можуть і посилати дані на шину, і приймати з неї дані. Внутрішня шина даних являє собою лінію двостороннього зв'язку. Варто пам'ятати, що по шині передаються слова даних, а не окремі біти. Так, у 16-розрядному мікропроцесорі всі пересилки по шині здійснюються групою з 2 байт (16 біт).

Лекція 5
КОМАНДИ МІКРОПРОЦЕСОРА
План

5.1. Структура команди мікропроцесора

Команда мікропроцесора – це двійкове слово, що, будучи «прочитаним» мікропроцесором, змушує його виконати визначені дії.

Довжина команди двійкового слова збігається з довжиною слова даних. Так, довжина слова команди 8-розрядного мікропроцесора дорівнює 8 біт, а 16-розрядного мікропроцесора – 16 біт і т.д. Проте команди можуть мати довжину, рівну не тільки одному, але також двом або трьом словам. Отже, довжина команди 8-розрядного мікропроцесора може бути 8, 16 або 24 біт.

Для виконання команда посилається в регістр команд, дешифратор і схеми керування, де вона ідентифікується, у результаті чого формуються сигнали, що спрямовуються в інші частини мікропроцесора. За допомогою цих сигналів виконуються операції, що наказуються командою.

Мікропроцесор завантажує команду в регістр команд протягом циклу вибірки. Протягом такого за ним циклу виконання мікропроцесор декодує команду і створює сигнали керування процесом виконання операцій цієї команди.

Команда складається з двох частин: *коду операції (КОП)* і *адреси.* Код операції повідомляє мікропроцесору, що робити; адреса вказує місце розташування даних, що беруть участь в операції. Якщо довжина команди складає два або три слова, то перше з них – це код операції, а друге і третє – адреса.

5.2. Мнемонічна форма запису команд мікропроцесора

Команда мікропроцесора – це двійкове число. Але навіть 1-байтове двійкове число важко запам’ятати.

Для позначення команд використовується мнемонічне позначення – скорочений запис назви команди. Для цієї мети зазвичай використовуються три букви назви операції, виконуваної командою.

Наприклад, мнемонічне позначення команди очищення має такий вид: CLA. Якщо мікропроцесор містить два акумулятори (A і B), то команди їхнього очищення можуть записуватися як CLA A і CLA B, де CLA – код операції, а A і B – адреси місця розташування даних, що обробляються. Якщо ж команда оперує числовими даними або адресами областей пам’яті, то доцільним є викорис-

тання чисел в адресній частині команди. Наприклад, код операції з мнемонічним позначенням JMP (JUMP–ПЕРЕХІД) потребує вказівки адреси переходу. Подібна команда може мати вид JMP 177756.

5.3. Неявна адресація

1-байтова команда 8-розрядного мікропроцесора – це одна з 256 різноманітних комбінацій 8 біт, що утворюють машинне слово (байт). Такої кількості різноманітних команд достатньо для аналізованого нами мікропроцесора. Проте він оперує 65 536 областями пам'яті, для адресації якої адресна частина команди повинна бути більша тієї, що може надати 1-байтова команда.

1-байтові команди не адресуються до даних, розташованих у пам'яті; вони оперують даними, завантаженими в регістр, реєстрову пару або даними, збереженими в області пам'яті, адреса якої знаходиться в реєстровій парі. Наприклад, 1-байтова команда пересилки даних із регістра A в регістр B складається з коду операції, адреси джерела даних (регістра A) і адреси приймача даних (регістра B). Адреси джерела і приймача зазначені в команді неявно; іноді говорять, що вони «вмонтовані» у команду. От чому така адресація називається *неявною.*

1-байтові команди виконуються швидше будь-яких інших команд. У випадку 1-байтової команди мікропроцесор витрачає на це два мікроцикли: один – на операцію вибірки, інший – на операцію виконання.

5.4. Безпосередня адресація

Цей засіб адресації неважкий для розуміння. Код операції команди з *безпосередньою адресацією* розміщується в першому байті. Відразу ж за кодом операції йдуть дані, що займають 1 або 2 байти. Ці дані беруться не з пам'яті, їх надає машині програміст при записі команди. Отже, при використанні даного засобу адресації не потрібно вказівка адреси пам'яті, необхідний тільки код операції, після якого записуються дані.

Безпосередня адресація здійснюється мікропроцесором за два мікроцикли: протягом першого циклу проводиться вибірка команди, протягом другого – її виконання. Операції, що задаються першим байтом команди (кодом операції), мікропроцесор виконує над даними, поданими її другим байтом. При використанні безпосередньої адресації в команді передбачається розміщення даних там же, де знаходиться сама команда.

5.5. Пряма адресація

Команди з *прямою адресацією* можуть мати довжину, рівну 2 або 3 байт. Перший байт призначений для коду операції, другий і, якщо є, третій – для адреси. Адреса вказує область пам'яті, у якому знаходяться дані. Спільне використання другого і третього байтів команди дозволяє адресуватися до будь-якої із 65536 областей пам'яті.

При неявній адресації адреса місця розташування даних «вмонтована» у команду, і програміст позбавлений можливості самостійно звертатися до даних по їхній адресі.

При безпосередній адресації дані вказуються в самій команді, слідуючи відразу за кодом операції. У цьому випадку програміст теж не може адресуватися до даних.

І тільки при прямій адресації в нього є така можливість, явно задаючи адресу необхідних даних.

Застосування прямої адресації пов’язане з використанням додаткового (у порівнянні з раніше розглянутими типами адресації) числа мікроциклів.

По-перше, мікропроцесору необхідно зробити вибірку коду операції команди. Після його декодування варто витягти з пам’яті ще 2 байти, що є значенням адреси місця розташування даних, що обробляються. При кожному додатковому звертанні до пам’яті потрібно ще один мікроцикл. Після отримання коду операції і байтів адреси йде етап виконання команди, на який затрачається четвертий мікроцикл. Отже, час виконання команд із прямою адресацією у два рази більший, чим команд із безпосередньою адресацією.

5.6. Непряма адресація

Більшість мікропроцесорів оперує ще одним засобом адресації до пам’яті, реалізованою командою довжиною тільки в одне слово. Така адресація називається *непрямою* або іноді *непрямою реєстровою.* Крім коду операції в такій команді вказується номер регістра, уміст якого – адреса місця розташування даних у пам’яті.

Непряма адресація зручна при звертанні до часто використовуваних областей пам’яті, і особливо в тих випадках, коли дані організовані у вигляді деякого списку або *файлу* (набору). Інакше кажучи, використання непрямої адресації дає найбільший ефект при записі і читанні областей пам’яті, що розміщені одна за одною. Прикладом може бути задача організації файлу даних.

5.7. Набори команд

У більшості випадків назви команд мікропроцесора в достатній мірі характеризують їхнє призначення.

Характеристики:

1. Назва.
2. Використовувані засоби адресації.
3. Мнемонічне позначення.
4. Розміщення команди в пам’яті.
5. Подання дій, виконуваних командою, за допомогою логічних символів.
6. Пояснення дій, виконуваних командою.
7. Довжину команди.
8. Вплив результату виконання команди на регістр стану мікропроцесора.

5.8. Команди пересилки даних

Використовуються для пересилки даних в різноманітні пристрої збереження інформації, а також із них.

Розрізняють команди *завантаження, пересилки та запису в пам'ять*.

LDA r, data	data → r	- завантаження регістра безпосереднє;
LDD r, address	M → r	- завантаження регістра пряме;
LDI A	M→A	- завантаження акумулятора непряме;
LRP B, data	M→C (M+1)→B	- завантаження регістрової пари безпосереднє;
MOV r1,r2	r2→r1	- пересилка з регістра в регістр;
STA A, address	A→M	- запис акумулятора в пам'ять прямий;
STI A	A→M	- запис акумулятора в пам'ять непрямий.

5.9. Арифметичні команди

З їх допомогою виконується обробка двох двійкових чисел за арифметичними правилами. Зводиться до операції ДОДАВАННЯ.

ADD r	A+r→A	- додавання з регістром;
ADD M, address	A+M→A	- додавання з пам'яттю пряме;
ADI M	A+M→A	- додавання з пам'яттю непряме;
ADD I	A+data→A	- додавання з даними безпосереднє;
ACD r	A+r+C→A	- додавання з регістром з переносом;
ACD M, address	A+M+C→A	- додавання з пам'яттю з переносом пряме;
ACI M	M+A+C→A	- додавання з пам'яттю з переносом непряме;
ACD I	data+A+C→A	- додавання з даними з переносом безпосереднє;
SUB r,	A–r→A	- віднімання з регістром;
SUB M, address	A–M→A	- віднімання з пам'яттю пряме;
SUI M	A–M→A	- віднімання з пам'яттю непряме;
SUB I, data	A–data→A	- віднімання з даними безпосереднє;
SCB r	A–r–C→A	- віднімання з регістром з переносом;
SCB M, address	A–M–C→A	- віднімання з пам'яттю з переносом пряме;
SCI M, address	A–M–C→A	- віднімання з пам'яттю з переносом непряме;
SCB I, data	A–data–C→A	- віднімання з даними з переносом безпосереднє.

Та інші.

5.10. Логічні команди

Використовуються для обробки даних що знаходяться в акумуляторі мікропроцесора.

AND r	A·r→A	- І над регістром і акумулятором;
AND M, address	A·M→A	- І над безпосередньою пам'яттю і акумулятором;
ANI M	A·M→A	- І над непрямою пам'яттю і акумулятором;
AND I, data	A·data→A	- І над безпосередніми даними і акумулятором;
OR r	A∨r→A	- АБО над регістром і акумулятором;
OR M, address	A∨M→A	- АБО над безпосередньою пам'яттю і акумулятором;
OR M	A∨M→A	- АБО над непрямою пам'яттю і акумулятором;
OR I, data	A∨data→A	- АБО над безпосередніми даними і акумулятором;

XOR r $A \oplus r \rightarrow A$ - ВИКЛЮЧАЮЧЕ АБО над регістром і акумулятором;

XOR M, address $A \oplus M \rightarrow A$ - ВИКЛЮЧАЮЧЕ АБО над безпосередньою пам'яттю і акумулятором;

XOR M $A \oplus M \rightarrow A$ -ВИКЛЮЧАЮЧЕ АБО над непрямою пам'яттю і акумулятором;

XOR I, data $A \oplus data \rightarrow A$ -ВИКЛЮЧАЮЧЕ АБО над безпосередніми даними і акумулятором;

CMA $\overline{A} \rightarrow A$ - ІНВЕРСІЯ акумулятора;

CMP r A–r - ПОРІВНЯННЯ регістра з акумулятором;

CMP M, address A–M - ПОРІВНЯННЯ безпосередньої пам'яті з акумулятором;

CMI M A–M - ПОРІВНЯННЯ непрямої пам'яті з акумулятором;

CMP I, data A-data - ПОРІВНЯННЯ безпосередніх даних з акумулятором.

Та інші.

5.11. Команди переходу та виклику підпрограм

1. Перехід безумовний - JMP, address
2. Перехід якщо 0 - JZ, address
3. Перехід якщо не нуль - JNZ, address
4. Перехід якщо перенос - JC, address
5. Перехід без переносу - JNC, address
6. Перехід якщо мінус - JN, address
7. Перехід якщо не мінус - JNN, address
8. Виклик підпрограми безумовний - CAL, address
9. Виклик підпрограми якщо 0 - CZ, address
10. Виклик підпрограми якщо не нуль - CNZ, address
11. Виклик підпрограми якщо перенос - CC, address
12. Виклик підпрограми без переносу - CNC, address
13. Виклик підпрограми якщо мінус - CN, address
14. Виклик підпрограми якщо не мінус - CNN, address
15. Повернення із підпрограми - RET

Лекція 6
ПАМ'ЯТЬ
План

6.1. Пам'ять мікропроцесорних систем
6.2. Типи ПЗП
6.3. Прямий доступ до пам'яті
6.4. Адреси пам'яті

6.1. Пам'ять мікропроцесорних систем

Для архітектури сучасних мікропроцесорів характерна наявність єдиного адресованого простору пам'яті, що називається *основною пам'яттю.* Принаймні якась частина основної пам'яті повинна являти собою *оперативну пам'ять,* тобто пам'ять із можливістю як читання областей пам'яті, так і запису в них інформації.

При роботі з пам'яттю часто говорять про *час доступу до пам'яті* і *часу циклу пам'яті.* Обидва параметри є характеристиками *швидкодії* пам'яті.

Один із різновидів параметра часу доступу вказує на те, який час необхідно для переведення інформації з пам'яті на шину даних після адресації потрібної області пам'яті. Вона називається *часом доступу при читанні або часом читання даних із пам'яті.* Інший різновид зазначеного параметру – *час доступу при запису – це* час, необхідний для запису даних у область, що адресується.

Час доступу визначається організацією пам'яті та швидкістю роботи схем, на яких побудована пам'ять.

Швидкодія пам'яті характеризується, крім того, часом циклу. *Час циклу* – це найменший інтервал часу, що може мати місце між двома звертання до пам'яті. Він залежить не тільки від характеристик, що властиві пам'яті, але і від інших параметрів системи.

При описі пам'яті мікропроцесорних систем часто використовується така її характеристика: *енергозалежна* або *енергонезалежна пам'ять.* У першій дані при порушеннях у роботі системи живлення дані не знищуються, а в другій – знищуються.

Для реалізації напівпровідникової пам'яті застосовуються два основних види технології, ті ж самі, що і для виготовлення інших цифрових інтегральних схем: біполярна і КМОН (CMOS) (комплементарна структура метал-оксид-напівпровідник) технології.

На сьогоднішній день для мікропроцесорних систем найбільше характерно використання пам'яті на КМОН - транзисторах.

Існують два засоби побудови інтегральних схем пам'яті на КМОН - технології, відповідно до яких пам'ять на КМОН – структурах може бути *статичною* або *динамічною.*

Статична пам'ять простіша відносно організації. Керувати роботою статичної пам'яті легше. Інтегральні схеми, які застосовуються для побудови динамічної пам'яті, дещо дешеві, але до складу такої пам'яті входить багато допоміжних інтегральних схем. Крім того, уміст динамічної пам'яті необхідно періодично регенерувати.

Комірка статичної пам’яті являє собою тригер. Цей тригер може бути встановлений або в стан 1, або в стан 0. Подібні осередки пам’яті об’єднуються в матричну структуру, розміщаються по рядках і стовпчиках.

Статичні пристрої, що запам’ятовують, є, мабуть, найбільше поширеним видом пам’яті мікропроцесорних систем. Більшість статичних пристроїв, що запам’ятовують, реалізується на основі КМОН -технології, але існують і статичні пристрої, що запам’ятовують, на біполярних транзисторах.

Комірка динамічної КМОН-пам’яті простіша статичної. За допомогою вдвічі меншої кількості транзисторів, чим в комірці статичної пам’яті, реалізовані ті ж функції. Комірка динамічної КМОН-пам’яті будується не у вигляді тригера. Збереження одиниці двійкової інформації здійснюється шляхом збереження заряду, що відповідає логічній 1, протягом декількох мілісекунд. Після закінчення цього часу необхідно виконати перезапис даних в комірку. Відсутність заряду на конденсаторі відповідає збереженню логічного 0.

Повторний запис даних в комірку динамічної пам’яті називається *регенерацією* даних. Пристрої динамічної КМОН-пам’яті постачаються так називаною логічною схемою регенерації. Ця схема автоматично здійснює звернення до кожного стовпчика пам’яті з інтервалами в декілька десятих долей мілісекунди.

ОЗУ:

1. Адресні лінії.
2. Вибір кристала (за рахунок однорозрядності реальних чіпів).
3. Лінії введення-виведення даних.
4. Лінія, що визначає читання або запис.

Кожна з аналізованих схем пам’яті має вхід «Запис». На цей вхід подається сигнал, коли необхідно записати дані в адресовану комірку пам’яті. Якщо ж сигнал на цей вхід не поданий, схема перебуває в режимі читання.

За рахунок дворазового використання адресних ліній (для старших і молодших розрядів) у всіх цих ОЗУ забезпечуються необхідні діапазони адресації, що складають відповідно:

6 адресних входів $2^6 \times 2^6 = 2^{12} = 4\,096$;

7 адресних входів $2^7 \times 2^7 = 2^{14} = 16\,384$;

8 адресних входів $2^8 \times 2^8 = 2^{16} = 65\,536$.

Отже, має місце така послідовність дій при читанні:

1. Мікропроцесор поміщає на адресні лінії пам’яті інформацію, допустиму для адресації.

2. Пристрій керування мікропроцесора виробляє сигнал «Читання».

3. Вміст адресованої комірки пам’яті надходить на зовнішні виводи кристала пам’яті.

4. Вихідні дані кристала пам’яті подаються через буфер на шину даних мікро-.

5. Мікропроцесор приймає ці дані через порт шини даних і пересилає їх по місцю призначення.

Послідовність дій при записі така ж, змінюються лише назви сигналів.

1. Мікропроцесор поміщає на адресні лінії пам’яті інформацію, припустиму для адресації.

2. Пристрій керування мікропроцесора виробляє сигнал «Запис».

3. Мікропроцесор виводить дані через порт шини даних.

4. Інформація із шини даних мікро-обчислювальної системи надходить через буфер на входи даних мікросхеми пам'яті.

5. Дані записуються в обрану область пам'яті.

6.2. Типи напівпровідникових ПЗП

Постійний запам'ятовуючи пристрій (ПЗП) – це така пам'ять, інформація в яку, будучи записана, зміні не підлягає. Часто застосування ПЗП в мікропроцесорної системі викликано тою обставиною, що система завжди виконує ту саму програму. Команди такої програми зберігаються в ПЗП. При використанні пам'яті цього типу не виникає проблема енергозалежності, пов'язаної з застосуванням напівпровідникових ОЗП.

Існують чотири типи ПЗП різноманітного призначення.

1. ***ПЗП** з масковим програмуванням (MROM)* – пам'ять, у якій інформація записана разом і назавжди в процесі виготовлення напівпровідникових інтегральних схем., тому що набір бітів, що підлягає запам'ятовуванню, фіксується в ході технологічного процесу з використанням шаблонів.

Діоди заводського типу в матрицях. Плівки в матрицях

2. Далі перейдемо до вивчення *програмованого постійного запам'ятовуючого пристрою* (***ППЗП**(PROM)*), Комбінація бітів, що вводиться в ППЗП, може бути задана користувачем. Програмування ППЗП – це одноразово виконувана операція, тобто інформація, колись записана в ППЗП, згодом змінена бути не може. Якщо необхідно зберегти в ППЗП іншу комбінацію бітів, потрібно використовувати інші мікросхеми ППЗП.

Перепалюються діоди.

3. Стираючий *програмований постійний запам'ятовуючий пристрої* (***СППЗП**(EPROM)*), при роботі з ним користувач може запрограмувати його, потім стерти записану інформацію і знову запрограмувати ті ж самі мікросхеми. Будучи значно більш дорогими, чим ПЗП і ППЗП, СППЗП дають користувачу можливість у міру потреби замінити інформацію, що утримується в СППЗП.

Властивості напівпровідників змінювати опір під дією струму й УФ-випромінювання.

4. *Електрично змінюваний постійно запам'ятовуючий пристрій* (***ЕЗПЗП*** (*EEPROM*)), програмування і зміна вмісту якого здійснюються за допомогою електричних засобів. На відміну від СППЗП для стирання інформації, збереженої в ЭИПЗУ, не потрібно спеціальних зовнішніх пристроїв.

Деякі властивості напівпровідникових приладів залежать від часу дії сигналів.

6.3. Прямий доступ до пам’яті.

Всі пересилки даних, що мають місце при їхньому введенні і виведенні відбувалися в ході виконання програм. Це дуже повільний процес, оскільки виконуються ці програми мікропроцесором.

Існує необхідність наявності *прямого доступу до пам’яті (ПДП)*. Прямий доступ до пам’яті, це один із засобів організації швидкої пересилки даних при обміні інформацією з пам’яттю.

У більшості випадків ПДП реалізується шляхом використання спеціального контролера ПДП. На час виконання ПДП мікропроцесор повинен бути позбавлений можливості звертання до пам’яті. Для того, щоб почати реалізацію режиму ПДП, мікропроцесор завантажує в контролер ПДП вміст деякого зовнішнього регістра, що являє собою початкову адресу файла даних. Крім того, в інший регістр (лічильник пересилок), що знаходиться в контролері ПДП, завантажується кількість байтів, що підлягає пересилці. Потім мікропроцесор переходить у режим заборони подачі адрес і даних по шинах і передає керування пам’яттю контролеру ПДП. Контролер ПДП послідовно подає на адресну шину пам’яті мікропроцесорної системи адреси і виробляє сигнали керування читанням і записом. При пересилці кожного байту здійснюється від’ємне збільшення вмісту регістра-лічильника. Коли вміст лічильника стає рівним 0, у зовнішній пристрій подається повідомлення, що пересилка даних завершена. Будучи вузько спеціалізованим пристроєм, контролер ПДП виконує всі ці дії дуже швидко. На сьогоднішній день час ПДП-пересилок практично наближається до часу циклу пам’яті. Закінчивши пересилку даних у пам’ять або з неї, контролер ПДП знову передає керування мікропроцесору.

6.4. Адреси пам’яті

Логічні адреси пам’яті (logical memory addresses) – це адреси, що використовуються програмами. Цей термін може використовуватися незалежно від того, чи працює програма в реальному або захищеному режимі. Цей термін просто позначає числа, що визначають, куди вказує програма в основній пам’яті.

Сегментована адреса пам’яті (segmented memory address) – означає адресу в програмі реального режиму. Вона вказується парою 16-розрядних чисел, що розділяються двокрапкою, наприклад 1A35:0043. (Обидва числа шістнадцяткові). Перше число позначає сегмент, і його значення, помножене на 16, додається до другого числа, що називається зсувом, для одержання реальної, фізичного адреси пам’яті.

У захищеному режимі те, що називалося сегментованою адресою, називається *віртуальною адресою пам’яті* (virtual memory address). Це теж пара шістнадцяткових чисел, розділених двокрапкою. Проте в цьому випадку перше число (також 16-бітне) називається селектором і вказує, який сегмент необхідно використовувати. Друге число – зсув, може бути 32-бітним числом.

Як сегментовані, так і віртуальні адреси пам’яті перекладаються з двох чисел в одне, що називається *лінійною адресою пам’яті* (linear memory address). Його максимальний розмір у PC поки не перевищує 36 біт, проте ця можливість поки не була реалізована в жодній операційній системі PC.

Починаючи з 386, усі процесори Intel серії x86 і усі їхні клони мають умонтований *сторінковий механізм пам'яті* (memory paging mechanism), що може бути або включений, або виключений, за вимогою програмного забезпечення. Якщо він відключений, то на адресні входи мікропроцесора подається лінійна адреса, що і стає фізичною адресою пам'яті.

Якщо ж сторінковий механізм включений, то лінійна адреса пам'яті перекладається на адресу фізичної пам'яті за допомогою звертання до двох таблиць в основній пам'яті. Назви цих таблиць – *каталог таблиць сторінок* (page directory table) і *таблиця сторінок* (page table).

Старші 10 біт лінійної адреси (біти з 22 по 31) вибирають рядок у каталозі таблиць сторінок. Цей рядок містить число, що вказує на місце в пам'яті, де можна знайти відповідну таблицю сторінок. Інші 10 біт (біти з 12 по 21) вибирають рядок у цій таблиці сторінок. Число, що там *знаходиться,* указує сторінку (page frame), що являє собою область адресного простору пам'яті довжиною 4Кб. Останні 12 біт лінійної адреси (біти з 0 по 11) указують на конкретне місце усередині цієї сторінки. Запис в таблицях сторінок і в каталозі таблиць сторінок, крім того, містять і деяку додаткову інформацію, відповідно, про сторінку і таблицю сторінок, на які вони вказують.

PC може працювати в двох режимах – реальному і захищеному.

Реальний режим – коли програмі що виконується доступна вся пам'ять. Операційна система реального режиму DOS. Їй доступна вся пам'ять, але вона може адресувати тільки перший мегабайт пам'яті.

Захищений режим – коли програмам що виконуються виділяються визначені, не доступні для інших програм області пам'яті. (Windows NT, OS/2, Unix).

Windows 95/98 – оболонки, що реалізують захищений режим на реальному режимі DOS.

У реальному режимі пам'ять адресується спеціальним представленням, що називається адресацією сегмент:зсув (segment:offset).

Принцип адресації сегмент:зсув означає, що кожне посилання в програмі на місце в пам'яті використовує два 16-розрядних числа: *номера сегмента* (segment value) і *зсуви* (offset). Номер сегмента, помножений на 16, додається до зсуву для отримання реальної фізичної адреси. Саме тому в реальному режимі PC може адресувати тільки біля 1 Мб пам'яті. Це розмір адресного простору реального режиму. Сучасні PC запрограмовані на перехід у захищений режим на початку процесу завантаження, у якому вони можуть адресувати усю встановлену пам'ять.

У захищеному режимі переклад логічної адреси у фізичну декілька складніше. Потрібно декілька кроків для об'єднання значення *селектора* (ця назва сегмента в захищеному випадку) із зсувом, і передача адреси, що утворилася, через таблиці перекладу сторінок, перед тим, як утвориться фізична адреса.

Важливою ідеєю даного поділу є те, що кожний PC на початку завантаження має адресний простір усього лише в 1 Мб.

Працюючи в захищеному режимі, CPU (не нижче 386) має доступ до повного адресного простору в 4 Гб, а для цього необхідна 32-розрядна шина адреси.

У захищеному режимі номер сегмента (також 16-бітне число) називається селектором, а зсув розширений із 16-бітного числа до 32-бітного.

Значення селектора саме по собі не є частиною адреси в пам’яті, а являє собою посилання на таблицю даних, називану *таблицею дескрипторів сегментів* (segment descriptor table). Для всіх програм, виконуваних на PC у визначений момент часу, існує одна таблиця, яка називається *глобальною таблицею дескрипторів* (Global Descriptor Table, GDT), і ще по одній таблиці для кожної програми; ця таблиця називається *локальною таблицею дескрипторів* (Local Descriptor Table, LDT)). Частина інформації в цій таблиці дескрипторів складає базова адреса даного сегмента, що може знаходитися де завгодно в 4Гб пам’яті, що адресується CPU.

Інша частина таблиці дескрипторів сегмента вказує його розмір і включає біт, що називається *бітом гранулярності.* Якщо цей біт скинутий, максимальний розмір сегмента складає 1 Мб. Розмір може бути будь-яким числом, аж до максимуму. Якщо ж цей біт установлений, число довжиною в 20 біт, що позначає розмір, множиться на 4096. У цьому випадку максимальний розмір сегмента дорівнює повному розміру адресного простору (4 Гб), і цей розмір виражається цілим числом, кратним 4 Кб.

Операційна система реального режиму може, хоча це зовсім не обов’язково, установити біт гранулярності і зробити базову адресу сегмента рівним нулю для всіх сегментів. Якщо це зробити, кожна програма «побачить» повний 32-бітний адресний простір.

Засобом захистити одну програму від іншої є забезпечення невидимості адресного простору пам’яті однієї програми з іншою. А це потребує, щоб всі програми використовували сегменти, менші по розміру, ніж повний адресний простір.

Лекція 7
КОНТРОЛЕР ПРЯМОГО ДОСТУПУ ДО ПАМ'ЯТІ

План

7.1. Принципи роботи контролера ПДП
7.2.Типи передач
7.3. Опис внутрішніх регітрів ПДП

7.1. Принципи роботи контролера ПДП

Контролер прямого доступу до пам'яті Intel 8237A (КР1810ВТ37А) (ПДП, DMA – Direct Memory Access) забезпечує високошвидкісний обмін даними між пристроями вводу-виводу та ОЗУ без використання центрального процесора, що дозволяє звільнити процесор для виконання обчислень паралельно з обміном та незалежно від нього. Найбільш часто можливості ПДП використовуються при роботі з дисковими накопичувачами, однак реалізоване використання ПДП адаптерами накопичувачів на магнітній стрічці та поруч інших пристроїв. Значні переваги дає використання ПДП у процесі обміну з пристроями, що приймають або передають дані досить великими порціями з високою швидкістю.

Контролер має 4 незалежних канали, кожен з яких може обслуговувати один периферійний пристрій.

У роботі ПДП розрізняють 2 основні цикли: цикл очікування (Idle cycle) та активний цикл (Active cycle). Кожен цикл поділяється на ряд станів, що займають за часом одиницю часу (тік). З циклу очікування контролер може бути переведений в стан програмування (Program Condition) шляхом подачі на вхід RESET сигналу високого рівня, тривалістю не менше 300 нс і слідуючою за ним подачею сигналу низького рівня (рівень 0) на вихід CS (Chip Select). У стані програмування контролер буде знаходитись до того часу, поки на виході CS зберігається сигнал низького рівня. У процесі програмування контролера задаються:

- початкова адреса пам'яті для обміну;
- зменшене на одиницю число байтів, що передаються;
- напрям обміну,

а також встановлюються необхідні режими роботи (дозволити чи заборонити циклічну зміну пріоритетів, автоініціалізацію, задати напрямок зміни адреси при обміні і т.д.).

Завантаження 16 розрядних регістрів контролера здійснюється через 8-розрядні порти вводу-виводу. Перед завантаженням першого (молодшого) байту повинен бути скинутий (очищений) тригер-заглушка (тригер перший / останній, First/Last flip-flop), який змінює свій стан після виведення в порт першого байту і таким чином дає можливість наступній команді виводу в той же порт завантажити старший байт відповідного регістра.

Запрограмований канал повинен бути демаскованим (біт маски каналу встановлюється при цьому в 0), після чого він може приймати сигнали «Запит на ПДП», який генерується тим зовнішнім пристроєм, що обслуговується через цей канал. Сигнал «Запит на ПДП» може бути також ініційованим установкою

в 1 біта запиту даного каналу у регістрі запитів контролера. Після появи сигналу запиту контролер входить в активний цикл, у якому виконується обмін даними. Обмін може здійснюватися в одному з чотирьох режимів:

1. Режим одиночної передачі (Signle Transfer Mode). Після кожного циклу передачі контролер звільняє шину процесору, але зразу ж починає перевірку сигналів запиту і, як тільки виявляє активний сигнал запиту, ініціює наступний цикл передачі.

2. Режим блочної передачі (Block Transfer Mode). У цьому режимі наявність сигналу запиту потрібна лише до моменту видачі контролером сигналу «Підтвердження запиту на ПДП» (DACK), після чого шина не звільняється, аж до завершення передачі всього блоку.

3. Режим передачі за вимогою (Demand Transfer Mode). Даний режим є проміжним між двома першими: передача йде неперервно до тих пір, поки активний сигнал запиту, стан якого перевіряється після кожного циклу передачі. Як тільки пристрій не може продовжувати передачу, сигнал запиту скидається ним і контролер призупиняє роботу. Цей режим застосовується для обміну з повільними пристроями, що не дозволяють за своїми часовими характеристиками працювати з ПДП у режимі блочної передачі.

4. Каскадний режим (Cascade Mode). Режим дозволяє включити в підсистему ПДП більше одного контролера в тих випадках, коли недостатньо чотирьох каналів ПДП. У цьому режимі один із каналів ведучого контролера використовується для каскадування з контролером другого рівня. Для роботи в каскаді сигнал HRQ («Запит на захват») ведучого контролера подається на вхід DREG («Запит на канал ПДП») ведомого, а сигнал DACK («Підтвердження запиту») ведомого подається на вхід HDLA («Підтвердження захоплення») ведучого.

Така схема підключення подібна до підключенню ведучого (першого) контролера до мікропроцесора, з яким він обмінюється сигналами HRQ і HDLA.

7.2.Типи передач

Передача пам’ять-пам’ять (Memory-to-memory DMA). Використовується для передач блоку даних із одного місця пам’яті в інше. Вихідна адреса визначається в регістрах нульового каналу, вихідний – в регістрах першого каналу. Число циклів обміну (число байт мінус 1) задається в регістрі числа циклів каналу. Передача відбувається з використанням робочого регістру контролера в якості проміжної ланки для збереження інформації. При передачі пам’ять-пам’ять може бути заданий спеціальний режим фіксації адреси (Address hold), при якому значення поточної адреси в регістрі нульового каналу не змінюється, при цьому весь вихідний блок пам’яті заповнюється одним і тим же елементом даних, що знаходяться за заданою адресою.

Автоініціалізація (автозавантаження, Autoinitialization). Після завершення звичайної передачі, використаний канал ПДП маскується і повинен бути перепрограмований для подальшої роботи з ним. При автоініціалізації маскування каналу після завершення передачі не відбувається, а регістри поточної адреси та лічильник циклів автоматично завантажуються із відповідних регістрів із початковими значеннями. Таким чином для продовження (повторення) обміну

достатньо виставити сигнал запиту на ПДП по даному каналу.

Режим фіксованих пріоритетів. У цьому режимі канал 0 завжди має максимальний пріоритет, а канал 3 – мінімальний. Це означає, що будь яка передача по каналу з більш високим пріоритетом буде виконуватися раніше, ніж по каналу з більш низьким пріоритетом.

Циклічний зсув пріоритетів. Дозволяє уникнути «забивання» шини одним каналом при одночасній передачі по кількох каналах. Кожному каналу, по якому пройшла передача, автоматично присвоюється нижчий пріоритет, після чого право на передачу отримує канал з найвищим пріоритетом, для якого передача в даний момент можлива. Таким чином, якщо на початку роботи розподіл пріоритетів було звичайним (канал 0 – найвищий), і надійшли сигнали запрошення на ПДП по 1-му та 2-му каналах, то спочатку буде виконано передачу по першому каналу, тоді він отримає нижчий пріоритет (а канал 2, відповідно, вищий, так як зсув пріоритетів циклічний) і передача виконається по 2-му каналу, який потім отримає нижчий пріоритет, а вищий пріоритет отримає, відповідно, канал 3, який і буде мати пріоритет на передачу.

Стиснення часу передачі (Compressed transfer timing). У випадку, якщо часові характеристики швидкодії пристроїв, що обмінюються співпадають, ПДП може скоротити час виконання кожного такту передачі на 2 часових цикли за рахунок тактів очікування, що входять в кожний цикл передачі.

7.3. Опис внутрішніх регістрів ПДП

Контролер має 344 біти внутрішньої пам’яті, організованої у вигляді регітрів. Опис внутрішніх регістрів ПДП наведено в табл. 5.

Таблиця 5

Опис внутрішніх регістрів ПДП

Назва регістра	Розрядність (біт)	Число регістрів
Регістр початкової адреси (Base Address Register)	16	4
Регістр початкового лічильника циклів (Base Word Count Register)	16	4
Регістр поточної адреси (Current Address Register)	16	4
Регістр поточного лічильника циклів (Current Word Count Register)	16	4
Робочий регістр адреси (Temporary Address Register)	16	1
Робочий регістр лічильника циклів (Temporary Word Count Register)	16	1
Регістр стану (Status Register)	8	1
Регістр команд (Command Register)	8	1
Регістр режиму (Mode Register)	6	4
Робочий регістр (Temporary Register)	8	1
Регістр масок (Mask Register)	4	1
Регістр запитів (Request Register)	4	1

Регістр початкової адреси (Base Address Register). У цьому регістрі задається стартова адреса ОЗУ, з якої починається передача. Регістр містить 16 розрядів і визначає адресу всередині заданої сторінки пам'яті розміром 64К. Задання номерів сторінок пам'яті здійснюється через спеціальні сторінкові регістри (Page Registers), що підтримуються зовнішньою логікою. Кожен канал ПДП має свій регістр початкової адреси та сторінковий регістр. Таке розділення пам'яті на сторінки не дозволяє здійснити обмін з блоком пам'яті, що знаходяться на перетині двох сторінок. Кожна сторінка починається з сегментної адреси, кратної l000 h (0, l000h, 2000h, …, 9000 h).

Регістр початкового лічильника циклів (Base Word Count Register). В даному регістрі задається початкове число циклів передачі для програмованого каналу. Фактичне число переданих під час роботи ПДП елементів даних на одиницю перевищує задане число циклів, тобто, якщо ви задаєте 100 циклів передачі, а розмір елемента буде рівним 1 байту, то за сеансі обміну буде передано 101 байт інформації.

Регістр поточної адреси (Current Address Register). Початкове значення заноситься до цього регістру одночасно з регістром початкової адреси. Надалі в ході передачі значення поточної адреси автоматично збільшується або зменшується (конкретна зміна напряму задається при програмуванні в регістрі режиму). Якщо дозволена автоініціалізація, то після закінчення передачі в регістрі автоматично встановлюється значення з регістра поточної адреси.

Регістр поточного лічильника циклів (Current Word Count Register). Регістр містить поточне значення лічильника циклів (число циклів передачі, які залишилися). Відображене в ньому число циклів завжди на одиницю менше числа ще не переданих елементів даних, так як зміна значень у цьому регістрі відбувається в кінці циклу передачі, вже після фактичної передачі елемента даних, а кінець передачі фіксується в момент переповнення лічильника (зміна його значення на 0FFFFh).

Регістр режиму (Mode Register). Даний регістр задає режими роботи свого каналу контролера. Кожен з чотирьох каналів ПДП має свій набір регістрів, описаних вище. Крім того, наявний наступний набір регістрів, спільних для всіх каналів.

Регістр команд (Command Register). Цей 8-бітний регістр керує роботою контролера. Він програмується, коли контролер знаходиться в стані програмування та очищається командами скидання «Reset» та «Master Clear».

Регістр стану (Status Register). Регістр відображає поточний стан запитів і передач по всіх чотирьох каналах. Біти 0 – 3 встановлюються в одиницю після завершення передачі по каналах 0–3 (біт 0-канал 0, біт 1-канал 1 і т.д.), якщо не заданий режим автоініціалізації, вони очищаються після команди скидання контролера та після кожної операції зчитування стану з регістру стану. Біти 4–7 вказують по якому з каналів 0 – 3 активний в поточний момент сигнал запиту на ПДП.

Регістр масок (Mask Register). Кожен біт цього 4-бітного регістру маскує/демаскує свій канал ПДП, при цьому значення 1 маскує канал, значення 0 демаскує канал і дозволяє прийом сигналу запиту по цьому каналу.

Регістр запитів (Request Register). Сигнал запиту на ПДП (DREQ) може бути, згенерований, як обслуговуючим пристроєм, так і програмно. Для програмного генерування сигналу запиту по одному з 4-х каналів ПДП необхідно встановити відповідний біт у 4-бітному регістрі запитів. Запит на ПДП може бути відмінений записом нульового значення в відповідний біт регістра. Біт запиту очищається автоматично при завершенні передачі по даному каналу. Всі біти запитів очищуються при скиданні контролера. Для того щоб сприймати програмні запити на ПДП, канал повинен знаходитися в режимі блокової передачі.

Робочий регістр (Temporary Register). Цей 8-розрядний регістр використовується для зберігання елемента даних, що передається в режимі фіксованої адреси при передачі пам’ять-пам’ять або для тимчасового зберігання байта, що передається при всіх інших режимах передачі.

Лекція 8
ПРОГРАМОВАНИЙ КОНТРОЛЕР ПЕРЕРИВАНЬ

План

8.1. Функції контролера переривань
8.2. Опис основних елементів програмованого контролера переривань
8.3. Режими роботи програмованого контролера переривань

8.1. Функції контролера переривань

Програмований контролер переривань (ПКП, Programmable Interrupt Controller, PIC) реалізує векторну систему переривань. Мікросхема Intel 8259A (КР580ВН59), а так само її модифікації 8259A-2 і 8259A-8, підтримує 8 рівнів переривань від восьми різних пристроїв.

Основні функції контролера:

- фіксація запитів на переривання від восьми зовнішніх джерел;
- програмне маскування запитів, що надходять;
- привласнення фіксованих або циклічно змінюваних пріоритетів входів контролера, на які надходять запити;
- ініціація виклику процедури опрацювання надходження апаратного переривання.

Кількість обслуговуючих зовнішніх джерел переривань може бути збільшено шляхом каскадування декількох контролерів.

До складу контролера входять:

- схема керування читанням / записом;
- схема керування;
- схема каскадування;
- регістр запитів на переривання;
- схема обробки пріоритетів;
- регістр стану;
- регістр маскування запитів на переривання.

ПКП може перебувати в двох основних станах: налагодження і обслуговування запитів на переривання. У стані налагодження контролер приймає керуючі слова ініціалізації (Initialization Command Words, ICW), в стані обслуговування – операційні керуючі слова (Operation Control Words, OCW).

Можливі кілька режимів обслуговування джерел переривань:

- режим фіксованих пріоритетів за рівнями переривань;
- два різних варіанти циклічного зсуву пріоритетів;
- режим автоматичного завершення обробки переривання;
- режим спеціального маскування;
- режим опитування пристроїв.

8.2. Опис основних елементів програмованого контролера переривань

Схема керування читанням / записом (Read / Write Control Logic). Основною функцією цього блоку є прийом команд від мікропроцесора і передача йому інформації про стан ПКП. Обмін з мікропроцесором здійснюється через спеціальний 8-розрядний буфер даних (Data Bus Buffer), який є інтерфейсом

між ПКП і шиною даних. До складу блоку входять регістри керуючих слів ICW і OCW. Схема керується входами CS, RD, WR і A0. Вхід CS (Chip select) відповідає за вибір мікросхеми. Низький рівень сигналу на вході CS дозволяє виконання обміну з ПКП. Низький рівень сигналу на вході WR (Write) дозволяє мікропроцесору виводити керуючі слова ICW і OCW для прийому їх ПКП. Низький рівень сигналу на вході RD (Read) дозволяє ВКП передати мікропроцесору інформацію про стан спеціальних регістрів IRR, ISR і IMR, які описані нижче.

Всі керуючі слова ICW і OCW приймаються контролером у вигляді 9-розрядних значень. Розряди 0–7 передаються через 8-розрядний буфер даних. Старший розряд (восьмий, вважаючи з нуля) носить назву A0 і встановлюється в 0 або 1 в залежності від того, через який з двох можливих портів введення-виведення (парний чи непарний) було передано керуюче слово. Якщо для виведення значення використовувався порт з парною адресою, A0 буде дорівнює 0, якщо використовувався порт з непарною адресою на одиницю більшою, ніж попередній, тоді A0 буде дорівнює 1.

Регістр запитів на переривання (Interrupt Request Register, IRR) обслуговується через входи IR0–IR7 контролера. Сигнал на одному входів IR0–IR7 – це запит на переривання відповідного рівня (0–7). Відповідно з сигналом запиту на переривання схемою керування встановлюється відповідний біт в регістрі IRR.

Регістр стану (регістр оброблюваних запитів (In-Service Register, ISR) описує в бітах 0–7 переривання яких рівнів (0–7) в даний момент опрацьовуються.

Регістр маскування запитів на переривання (Interrupt Mask Register, IMR) описує, переривання яких рівнів в даний момент замасковані. Одиничне значення біта в IMR вказує на те, що переривання відповідного рівня при появі запиту в IRR блокується.

Схема обробки пріоритетів (шифратор пріоритетів, Priority Resolver) визначає, переривання якого рівня в даний момент є найбільш пріоритетним для виконання.

Схема керування ПКП формує сигнал запиту на переривання, що надходить на вхід INT (запит на переривання) мікропроцесора. Якщо прапор IF регістра прапорів процесора дорівнює 1 (переривання дозволені), процесор відповідає сигналом по лінії INTA (підтвердження переривання), після чого скидається в 0 розряд IRR і встановлюється в 1 розряд ISR, що відповідають рівню оброблюваного переривання. Після отримання другого сигналу підтвердження від процесора по лінії INTA, ПКП передає на шину даних 8-бітовий номер переривання. Дана послідовність роботи схеми керування виконується при підключенні ПКП до системи з мікропроцесорами 8088/8086. При роботі з мікропроцесорами 8080/8085 послідовність роботи схеми керування дещо відрізняється від описаної вище. Основна відмінність полягає в тому, що процесору передається не тільки номер переривання, а й код команди процесора INT (переривання) – байт 0CDh.

Схема каскадування відповідає за роботу каскаду з декількох контролерів. При підключенні до ведучого контролера вихід INT кожного ведомого підключається до одного з входів IRQ0–IRQ7 ведучого. Далі цей сигнал переда-

ється ведучим на вхід INT процесора. Коли процесор повертає сигнал INTA, ведучий контролер не тільки встановлює біт в ISR і скидає біт в IRR, але і видає на свої виходи CAS0–CAS2 номер рівня переривання, до якого підключений ведомий, що послав запит на переривання. Сигнали по лінії CAS0–CAS2 приймаються всіма ведомими, проте обробляються тільки тим, який підключений до лінії IRQ з відповідним номером.

8.3. Режими роботи програмованого контролера переривань

Режим фіксованих пріоритетів (Fixed Priority, Fully Nested Mode). В даному режимі контролер знаходиться відразу після ініціалізації. Запити переривань мають жорсткі пріоритети від 0 до 7 (0 – вищий) і опрацьовуються відповідно до пріоритетів. Переривання з меншим пріоритетом ніколи не буде опрацьовано, якщо в процесі обробки переривань з більш високими пріоритетами постійно виникають запити на ці переривання.

Автоматичне зміщення пріоритетів (Automatic Rotation). В даному режимі дається можливість обробити переривання всіх рівнів без їх дискримінації. Наприклад, після обробки переривання рівня 4 йому автоматично присвоюється нижчий пріоритет, при цьому пріоритети для всіх інших рівнів циклічно зсуваються і переривання рівня 5 матимуть в даній ситуації вищий пріоритет і, отже, можливість бути обробленими.

Програмно-керований зсув пріоритетів (SpecificRotation). Програміст може сам передати команду циклічного зсуву пріоритетів ПКП, задавши відповідне керуюче слово. У команді задається номер рівня, якому потрібно присвоїти максимальний пріоритет. Після виконання такої команди пристрій працює так само, як і в режимі фіксованих пріоритетів, з урахуванням їх зсуву. Пріоритети зсуваються циклічно, таким чином, що якщо максимальний пріоритет був призначений рівню 3, то рівень 2 отримає мінімальний і буде оброблятися останнім.

Автоматичне завершення обробки переривання (Automatic End Of Interrupt, AEOI). У звичайному режимі роботи процедура обробки апаратного переривання повинна перед своїм завершенням очистити свій біт в ISR спеціальною командою, інакше нові переривання НЕ будуть оброблятися ПКП. У режимі AEOI потрібний біт в ISR автоматично скидається в той момент, коли починається обробка переривання потрібної процедурою обробки і від неї не потрібно видавати команду завершення обробки переривання (EOI). Складність роботи в даному режимі обумовлюється тим, що всі процедури обробки апаратних переривань повинні бути повторно вхідними, так, як за час їх роботи можуть повторно виникнути переривання того ж рівня.

Режим спеціальної маски (Special Mask Mode). Даний режим дозволяє скасувати пріоритетне впорядкування обробки запитів і обробляти їх у міру надходження. Після скасування режиму спеціальної маски попередній порядок пріоритетів рівнів зберігається.

Режим опитування (Polling Mode). В цьому режимі апаратні переривання не відбуваються автоматично. Поява запитів на переривання повинно визначатися зчитуванням IRR. Даний режим дозволяє також отримати від ПКП інформацію про наявність запитів на переривання і, якщо запити є, номер рівня з максимальним пріоритетом, той за яким є запит.

Список використаних джерел

1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. Москва : Вильямс, 2005. 1296 с.
2. Клакович Л. М., Левицька С. М., Костів О. В. Теорія алгоритмів: Навч. Посібник. – Львів: ЛНУ, 2008. – 140 с.
3. Глибовець М.М. Основи комп'ютерних алгоритмів. – К.: Вид. дім „КМ академія", 2003. – 452 с.
4. Гилмор Ч. Введение в микропроцессорную технику. Пер. с англ. - М: Мир, 1984. - 334 с
5. Внутреннее строение микропроцессора (Лекция) [Електронний ресурс]. – Режим доступу : http://mc-plc.ru/mps/vnutrennee-stroenie-mikroprocessora.htm
6. Алгебра логіки та проектування основних операційних вузлів [Електронний ресурс] : навч. посіб. / В. В. Булатецький, Л. В. Булатецька, О. М. Собчук; ВНУ ім. Лесі Українки. – Електронні текстові данні (1 файл: 3,67 Мбайт). – Луцьк : ВНУ ім. Лесі Українки, 2021. – 150 с. // Режим доступу : https://evnuir.vnu.edu.ua/handle/123456789/19364
7. Блохнин С. М. Шина ISA персонального компьютера IBM PC/AT. – М. : ПК "Сплайн", 1992 – 76 с.
8. Ю.С.Лукач, А.Е.Сибиряков Архитектура ввода-вывода персональных ЭВМ IBM PC. – Свердловск : Инженерно-техническое бюро,1990 [Електронний ресурс]. – Режим доступу : https://www.booksite.ru/fulltext/1/001/005/001/037.txt

УДК 004.655

Електронне мережне навчальне видання

В. В. Булатецький, Л. В. Булатецька

ЗАГАЛЬНІ ПРИНЦИПИ ФУНКЦІОНУВАННЯ ТЕХНІЧНИХ ЗАСОБІВ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

Текст лекцій
нормативної навчальної дисципліни
“Архітектура обчислювальних систем”