МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

С. Ю. Даншина

КОМП’ЮТЕРНІ ТЕХНОЛОГІЇ ДЛЯ ГІС-ДОДАТКІВ

Частина 1

ЗАГАЛЬНІ ПРИНЦИПИ ОРГАНІЗАЦІЇ КОМП’ЮТЕРНИХ ТЕХНОЛОГІЙ

Конспект лекцій

Харків «ХАІ» 2023

УДК [004.315+004.415.5](075.8)
Д19

Рецензенти: канд. техн. наук, доц. Б. М. Іващук,
канд. техн. наук О. І. Горб

Даншина, С. Ю.
Д19 Комп’ютерні технології для ГІС-додатків [Текст]: консп. лекцій. Ч. 1. Загальні принципи організації комп’ютерних технологій / С. Ю. Даншина. – Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2023. – 96 с.

ISBN 978-966-662-912-1

Розкрито поняття комп’ютерних технологій, розглянуто еволюцію та перспективи їх розвитку. Описано загальні принципи побудови комп’ютерів на рівні їх внутрішньої будови та глобальної взаємодії. Значну увагу приділено особливостям подання різних типів інформації в комп’ютерних технологіях. Наведено арифметичні й логічні основи перетворення інформації в ГІС-додатках. Теоретичні відомості поєднано з практичними прикладами, де як інструментарій аналізу даних і моделювання розглянуто систему комп’ютерного моделювання Scilab.

Для студентів, що вивчають курс «Комп’ютерні технології для ГІС-додатків», спеціальностей «Геодезія та землеустрій», «Науки про Землю».

Іл. 25. Табл. 25. Бібліогр.: 42 назви

УДК [004.315+004.415.5](075.8)

ISBN 978-966-662-912-1

ПЕРЕДМОВА

Сучасний стан розвитку суспільства характеризується різким збільшенням інформаційних потоків не тільки в засобах масової інформації, а й у сферах виробництва, науки, культури. Якщо донедавна ступінь розвитку суспільства визначався ступенем індустріалізації, то сьогодні його повною мірою характеризує ступінь інформатизації. А, отже, попит на інформацію й інформаційні послуги забезпечує поширення і все більш ефективне використання комп'ютерних технологій [1].

Під впливом нових технологій відбуваються докорінні зміни в будь-якій сфері життя сучасної людини, тому комп'ютери та комп'ютерні технології посідають головне місце. Оскільки без них неможлива ефективна робота в тій чи іншій галузі діяльності, важливим елементом підготовки сучасного фахівця стає опанування комп'ютерних технологій. Суттєвою особливістю комп'ютерних технологій, на відміну від інших, є висока динаміка розширення їх предметної області, що потребує своєчасного врахування науково-технічних досягнень в сфері інформаційних технологій [2].

Виходячи з указаних особливостей, головна мета курсу − надати студентам знання і забезпечити фундаментальність освіти в галузі інформаційних технологій, допомогти їм оволодіти методами застосування сучасних комп'ютерних технологій, а також навчити самостійно використовувати сучасний комп'ютерний інструментарій для вирішення фахових завдань. Завданням курсу «Комп'ютерні технології для ГІС-додатків» є вивчення основних принципів і правил побудови, організації сучасних комп'ютерних технологій, їх характеристик і правил взаємодії.

Перша частина курсу складається з чотирьох лекцій. У першій лекції акцентується увага на шляхах розвитку комп'ютерів і комп'ютерних технологій, адже без розуміння минулого не можна пояснити їх сучасний стан і майбутні перспективи. У другій лекції розкриваються питання архітектури комп'ютерів, описуються принципи їх побудови, їхньої структури та структури машинних команд, а також пояснюються фундаментальні правила взаємодії комп'ютерів у мережах. У третій лекції приділяється увага особливостям подання різних видів даних у сучасних ГІС-додатках. У четвертій лекції розглядаються арифметико-логічні основи перетворення інформації в ГІС-додатках. У додатку посібника наведено сучасний інструментарій для виконання інженерних розрахунків, моделювання та програмування – систему комп'ютерної математики Scilab, а також приклади її успішного використання під час вирішення завдань, розглянутих у теоретичній частині видання.

Після кожної лекції наведено тестові запитання та завдання для самостійного опрацювання матеріалу курсу та підготовки до практичних занять.

Лекція 1. КОМП’ЮТЕРИ ТА КОМП’ЮТЕРНІ ТЕХНОЛОГІЇ

План

1. Поняття «комп’ютерні технології».
2. Еволюція комп’ютерних технологій.
3. Перспективи розвитку комп’ютерних технологій.

1. Поняття «комп’ютерні технології»

Науки, які вивчають комп'ютерні технології, виникли нещодавно, тому їх однозначного визначення досі немає. Оскільки предметом наук є інформація, то їх часто плутають з інформатикою, комп'ютерними науками, прикладною математикою тощо. Але ці науки різні, тому розглянемо цю термінологію докладніше.

Базове поняття комп'ютерних технологій – інформація. Нині відомо близько 500 означень терміна «інформація», але жодного вичерпного немає. Більш того, інформація входить до переліку філософських категорій (поряд з поняттями речовини та енергії), тому її досить складно визначити у простих термінах.

Примітка *Феномен інформації виявився настільки неоднозначним, що його розуміння вважається однією з найскладніших проблем сучасності. Самі спроби сформувати поняття «інформація» та різні його трактування в працях учених і практиків змушують задуматися про незвичну роль інформації в сучасному житті суспільства.*

У підручнику з інформатики наведено таке означення поняття «інформація» [1]:

«Інформація – це факти, відомості, повідомлення, поняття, теоретичні положення, що зменшують початкову невизначеність про об'єкти матеріального світу».

Та ж сама плутанина є у визначенні інформаційної технології. Найчастіше цей термін асоціюється з поняттями комп'ютера й обчислювальної техніки. Наприклад, у ДСТУ 53113.1-2008 інформаційна технологія визначається як прийоми, способи та методи застосування засобів обчислювальної техніки під час виконання функцій збору, зберігання, оброблення, передавання та використання даних. Інтернет-енциклопедія «Вікіпедія» на запит видає таке означення: «Інформаційні або комп’ютерні технології (англ. information technology, IT) – широкий клас дисциплін і сфер діяльності, які належать до технологій створення, збереження, керування й оброблення даних із застосуванням обчислювальної техніки…». До цього типу означень можна віднести означення, прийняте ЮНЕСКО.

Виділяють такі напрями комп'ютерних технологій, як програмування, штучний інтелект, інформаційні системи, мікропроцесорна та обчислювальна техніка та ін. [1, 2]. Згідно з ДСТУ 53113.1-2008 інформаційна система – це організаційно впорядкована сукупність документів та інформаційних технологій з використанням засобів

обчислювальної техніки та зв'язку для реалізації інформаційних процесів.

Різновидом інформаційних систем є геоінформаційні системи (ГІС) – сукупність географічної і геопросторової інформації та комп'ютерних технологій, призначених для збору, зберігання, оброблення й використання просторово-координованих даних з метою їх аналізу, моделювання, візуалізації і поширення. Будь-яка ГІС складається з таких взаємозв'язаних компонентів [3, 4]:

– цифрові дані – географічна та геопросторова інформація, яку переглядають і аналізують з використанням апаратного і програмного забезпечення;

– апаратне забезпечення – комп'ютери, які використовують для зберігання, відображення й оброблення даних;

– програмне забезпечення – комп'ютерні програми, що виконуються на апаратному забезпеченні під час роботи з цифровими даними.

Програмне забезпечення, яке є частиною ГІС, називається ГІС-додатком [4].

Підсумовуючи, зазначимо, що невизначеність у встановленні суті комп'ютерних технологій, їх цілей і завдань, пов'язана з невизначеністю їх предмета – інформації. Однак предметна галузь науки «комп'ютерні технології» сформована і містить інформаційні процеси та системи, моделі, мови і технології їх опису, спрямовані на здобуття та застосування знань і прийняття на їх основі рішень з різних предметних галузей.

2. Еволюція комп’ютерних технологій

Фундаментом сучасного є минуле, знання про яке дає змогу краще зрозуміти сьогодення [2]. Тому розглянемо схематичний шлях розвитку комп‘ютерної техніки та технологій (рис. 1.1).

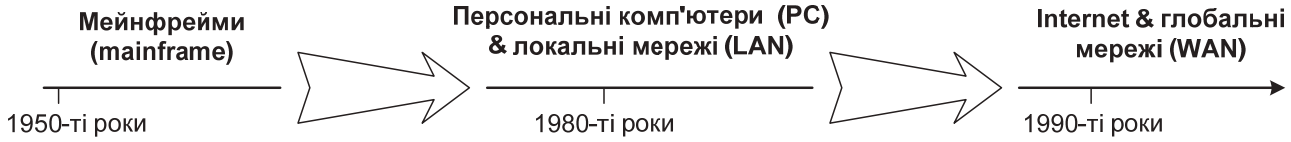


Рис. 1.1. Еволюція розвитку комп’ютерної техніки та технологій

В еволюції розвитку комп’ютерної техніки та технологій за останні роки умовно можна виділити (рис. 1.1): еру мейнфреймів; еру персональних комп’ютерів (PC) і локальних мереж (LAN); еру інтернету та глобальних мереж (WAN).

Мейнфрейм (mainframe) – високопродуктивний комп'ютер з великим обсягом оперативної та зовнішньої пам'яті, призначений для виконання інтенсивних обчислювальних робіт, здійснення критично важливих транзакцій та організації централізованих сховищ даних великої місткості.

Початком ери мейнфреймів вважають 1946 р. – рік створення першого електронного цифрового комп'ютера ENIAC. Надалі, незважаючи на високу вартість комп'ютерів першого покоління (близько мільйона

доларів США), їх виробництво стає комерційно вигідним, особливо у військових, банківських і державних установах для оброблення статистичних даних і грошових розрахунків. У цих галузях комп'ютер зміг замінити сотні службовців, багаторазово прискорити вирішення завдань, що дало суттєвий прибуток. Тому основним завданням комп'ютерів першого покоління стало накопичення і оброблення великих масивів структурованих даних. Більш універсальним програмно-керованим пристроєм для оброблення інформації, поданої в числовому вигляді, стала мала електронна обчислювальна машина, створена в Інституті електротехніки АН УРСР під керівництвом академіка Сергія Лебедєва та введена в експлуатацію в 1950 р. З появою в 1964 р. універсальної комп'ютерної системи IBM System/360 десятиліття з 1950-х до 1970-х стали вважати ерою мейнфреймів [5]. І хоча мейнфрейми вважають «застарілими» і час від часу з'являються повідомлення про переведення деяких промислових рішень (наприклад, ERP- і CRM-систем) з мейнфреймів на більш сучасні платформи, проте, як свідчить статистика, ці вихідці з 60-х, як і раніше, продовжують відігравати найважливішу роль у бізнесі найбільших компаній. Так, за прогнозами Gartner, останній мейнфрейм передбачалося вимкнути в 1993 р., але ринок мейнфреймів зростає: найближчим часом це зростання, за різними оцінками, становитиме 3 – 5 % на рік [6].

Як будь-який пристрій, мейнфрейми мають низку недоліків і переваг, серед яких зазначимо такі [6, 7]:

1. Зворотна сумісність: незалежно від версії операційної системи та дати виробництва мейнфрейм зобов’язаний підтримувати все, що працювало на попередніх версіях за минулі роки минулого століття. Наприклад, адресація пам'яті в мейнфреймі не 32- і 64-бітова, як у більшості сучасних комп'ютерів, а 24-, 31- і 64-бітова (тому що так відбувався розвиток); пристрої зберігання даних DASD вимірюють у треках і циліндрах; стрічки для виведення даних уже практично не використовуються, хоча є VTS (Virtual Tape Subsystem), які емулюють роботу зі стрічками та доступ до них.

2. Висока вартість і складність обслуговування, що вирішуються застосуванням нової елементної бази, яка значно знижує рівень енергоспоживання, спрощує вимоги до системи електроживлення й охолодження, зменшує габарити мейнфреймів при одночасному збільшенні продуктивності.

3. Машинні мови як основа взаємодії людини і мейнфрейму, що обумовлює їх доступність тільки для професіоналів. Це, наприклад, у період пандемії коронавірусу стало причиною багатьох проблем у США – для оновлення програмного забезпечення, яке використовують у системі зайнятості, терміново знадобилися фахівці з мови програмування COBOL (технології півстолітньої давності), але взяти їх не було звідки.

4. Централізоване оброблення даних і завдань, сформульованих

різними користувачами, що робить завдання будування інформаційних систем масштабу підприємства простішим і дешевшим, ніж у випадку застосування розподілених обчислень.

5. Здійснення окремими процесорами операцій «введення – виведення» за певними специфічними для пристрою командами, унаслідок чого на мейнфреймі можуть працювати більше користувачів (процесів), не впливаючи на продуктивність один одного. Подібний рівень паралелізму побутові комп'ютери змогли забезпечити лише з введенням у повсякденну діяльність хмарних обчислень.

6. Недотримання принципу відкритості систем (несумісність з іншими платформами), але при цьому мейнфрейми здатні підтримувати на одній машині сотні серверів з різними операційними системами, які призначено для вирішення різних завдань у гібридному хмарному середовищі.

7. Нативна підтримка десяткової арифметики, що є дуже зручним для фінансових додатків в умовах оброблення мільярдів транзакцій на день.

8. Надійність і відмовостійкість як ключові показники мейнфреймів. Це забезпечується: програмною надійністю (більша частина програм написана давно і перевірена десятиліттями роботи), фізичною надійністю (мейнфреми розроблялися за умови необхідності «переживати» різні природні катастрофи або мінімізувати втрати, пов'язані з ними) та кібербезпекою (скористатися будь-якою вразливістю неможливо, оскільки більшість нововведень із «звичайного» світу в мейнфрейми приходять із запізненням).

З 1948 р. почалася епоха транзисторної техніки, а з 1959 р. – інтегральних схем. У 1971 році американська фірма Intel випустила на комп’ютерний ринок мікропроцесор, що стало міцним поштовхом до прискорення розвитку і масового використання комп’ютерної техніки. Почалася ера персональних комп’ютерів і локальних мереж [2].

Персональний комп'ютер (PC) – будь-який повноцінний комп'ютер, що використовують як особистий.

Локальна мережа (LAN) – система розподіленого оброблення даних, яка об'єднує на фіксованій невеликій (діаметром до 10 км) території засоби передавання та оброблення інформації (PC, окремі пристрої тощо), орієнтовані на колективне використання загальномережних ресурсів – апаратних, інформаційних, програмних [8].

Перший персональний комп'ютер, доступний пересічному споживачу завдяки відносно невисокій вартості, – Altair, розроблено компанією MITS, США, на основі мікропроцесора фірми Intel у 1975 р. У 1981 р. компанія IBM представила перший PC IBM, побудований на принципі модульної архітектури, який став стандартом галузі й дав поштовх до розвитку цілої індустрії «сумісних з IBM» пристроїв, програмного забезпечення й аксесуарів. Розробляючи цей PC, IBM доручила іншим компаніям виготовляти його компоненти: мікропроцесор створювала Intel, операційну систему – DOS (дискова операційна система) – розробляла компанія

Microsoft. До речі, першу мову програмування Altair BASIC для першого PC також розробляла ця компанія, тільки тоді її назва була Micro-Soft [9, 10].

У 1976 р. Дж. Мерфі (John Murphy), інженер корпорації Datapoint, розробив технологію ARCnet, яку анонсував у 1977 р. ARCnet – комп'ютерна мережа прикріплених ресурсів, яка визначала протокол під'єднання до локальної мережі комп'ютерів будь-якого типу. ARCnet стала першою відкритою для придбання LAN, що давала змогу вузлам оброблення спілкуватися між собою для обслуговування файлів і виконання розподілених обчислень. Паралельно з цим у 1973 р. Р. Меткалф (Robert Metcalfe), розробник фірми Xerox, представив опис експериментальної мережі Ethernet, строге теоретичне підґрунтя якої з'явилося на початку 80-х рр. А в 1983 р. розроблений Інститутом інженерів електротехніки та електроніки (IEEE – Institute of Electrical and Electronics Engineers) сумісно з Xerox, DEC і Intel стандарт Ethernet було затверджено як стандарт IEEE 802.3. Незважаючи на відсутність стандартів, ARCnet у 1980 – 1990 рр. була дуже популярною та серйозно конкурувала з Ethernet. Багато компаній, наприклад Datapoint, Standard Microsystems, Xircom та ін., виробляли апаратуру для мережі цього типу, але зараз її майже не виробляють. Серед основних переваг ARCnet порівняно з Ethernet можна назвати точно визначений час доставки пакета даних, відому пропускну здатність мережі, надійність зв'язку, простоту діагностики, порівняно низьку вартість адаптерів. До найбільш істотних недоліків мережі належить низька швидкість передавання інформації – 2,5 Мбіт/с. Сьогодні технологія Ethernet займає 95 % всього ринку мережних технологій. Після її поширення як технології для створення LAN ARCnet знайшла застосування в вбудованих системах [8, 11].

Характерною рисою цього етапу розвитку комп'ютерних технологій є те, що обчислювальне навантаження розподіляється між постачальниками послуг (сервісів) – серверами і замовниками послуг – клієнтами за запитом користувачів з персонального комп'ютера. При цьому [8, 11]:

– розробляють операційні системи, які реалізують інтерактивний режим взаємодії декількох користувачів;

– широко впроваджують архітектуру «клієнт-сервер», яка знижує вимоги до комп'ютерів споживача, на яких встановлено клієнт;

– спрощується контроль повноважень на сервері, що дає доступ до даних лише тим клієнтам, які мають відповідні права;

– код програми-сервера не дублюється програмами-клієнтами;

– змінюються критерії ефективності оброблення даних: основними стають людські ресурси з розроблення та супроводу програмного забезпечення;

– з'являються вразливості в роботі мережі, тому що непрацездатність сервера може зробити непрацездатною всю обчислювальну мережу;

– стають потрібними фахівці, що підтримують роботу мережі – системні адміністратори.

Технологія LAN почала швидко розвиватися, а об'єднання локальних мереж у 1990-х рр. сприяло розвитку технології глобальних обчислювальних мереж (WAN). У цей самий час швидкий розвиток і популяризація технології інтернету вивели використання комп'ютерних технологій у нову еру, яка характеризується розвитком мережного оброблення даних і комунікаційних систем [11].

Глобальна комп'ютерна мережа (WAN) – це сукупність віддалених на велику відстань комп'ютерів-вузлів, об’єднаних для загального використання світових інформаційних ресурсів, сумісна взаємодія яких забезпечується комунікаційною мережею передавання даних і спеціальними програмами мережної операційної системи. Прикладами WAN є Internet, Bitnet, DECnet та ін.

Основні етапи розвитку WAN на прикладі розвитку інтернету наведено на рис. 1.2.

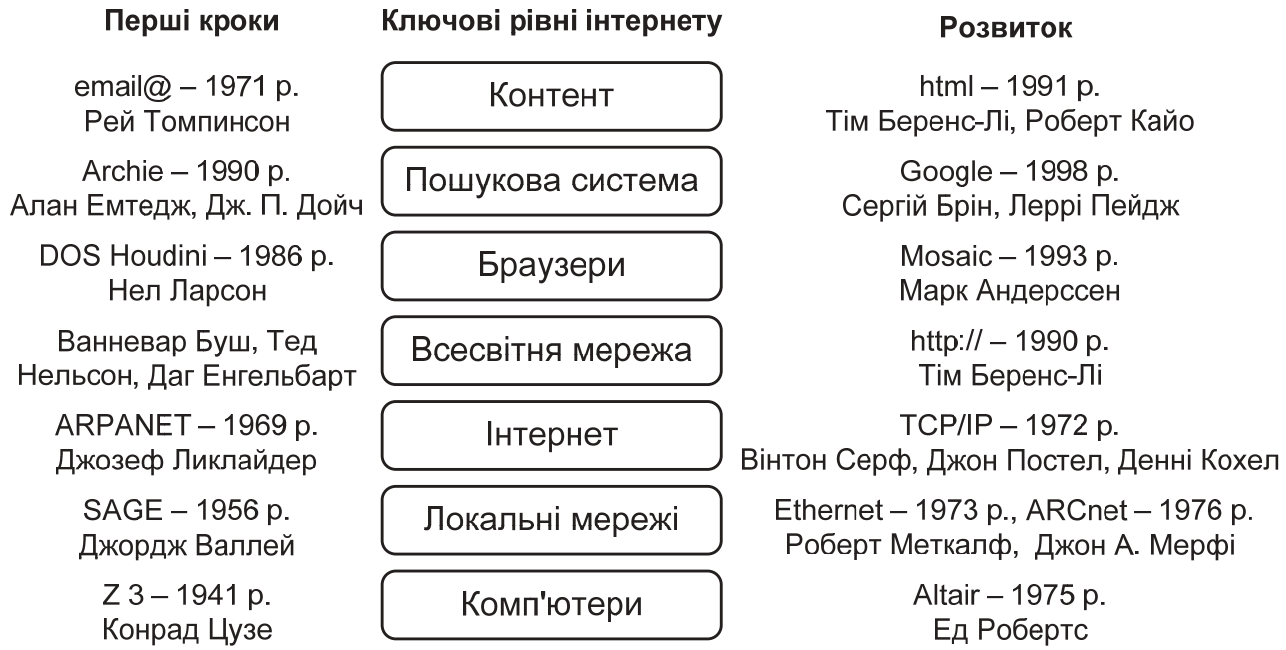


Рис. 1.2. Етапи розвитку інтернету
(за даними ресурсу https://uk.wikipedia.org/wiki/)

«Спілкування» комп'ютерів у мережі реалізовано за допомогою спеціальної мови – протоколу, який визначає правила передавання даних між вузлами комп'ютерної мережі. В інтернеті цю систему протоколів називають стеком протоколів TCP/IP, яка встановлює [11]:

– єдиний спосіб під’єднаня до WAN окремого комп'ютера або локальної мережі;

– єдині правила передавання даних;

– єдину систему ідентифікації комп'ютерів у мережі (мережну адресу).

При цьому WAN на відміну від LAN розраховані на необмежену кількість абонентів і використовують зазвичай не надто якісні канали

зв'язку з відносно низькою швидкістю передавання даних, а отже, механізм керування обміном у них не може бути швидким. Також з урахуванням того, що основною метою створення WAN була стійкість до часткових пошкоджень, тут використовують технологію децентралізованого оброблення даних.

Інтенсивний розвиток технологій, де основним предметом праці є інформація, стає наслідком зростаючого попиту на нові знаряддя праці – комп'ютери. Сьогодні комп'ютеризація – один із головних напрямів науково-технічного прогресу. Кількість та якість вироблених комп'ютерів, ступінь насиченості різних галузей господарювання обчислювальною технікою є одними з основних критеріїв економічного та воєнного потенціалу країн світу. За таких умов постійне збільшення кількості інформації, що обробляють і передають комп'ютерні технології, змушує задуматися про їх майбутнє [8, 12].

3. Перспективи розвитку комп’ютерних технологій

Традиційно, в сучасних умовах перспективи розвитку комп'ютерних технологій визначають за емпіричним законом про подвоєння числа транзисторів, якій запропонував у 1965 р. Г. Мур – один із засновників компанії Intel. Він помітив закономірність: нові моделі мікропроцесорів розроблялися через відносно однакові періоди часу (18 – 24 міс.), при цьому кількість транзисторів у них збільшувалася щоразу приблизно вдвічі. «Якщо така тенденція продовжиться, – припустив Мур, – то потужність комп'ютерів експоненціально зросте протягом відносно короткого проміжку часу, адже їх швидкодія залежить від кількості транзисторів у процесорах» [12].

Пізніше математичне трактування закону сформували так [13]:

$$N(y) = N_0 2^{\frac{y}{yy}}, \qquad (1.1)$$

де N_0 – кількість транзисторів у мікропроцесорі за рік, умовно прийнятий за початок відліку; N – кількість транзисторів у мікропроцесорі в період, що аналізується; y – кількість років, що минули з початку відліку, за які кількість транзисторів збільшилася; yy – строк (у роках і частках року), за який кількість транзисторів зростає вдвічі.

Початком відліку вважають 1965 р., коли технологія виробництва давала змогу інтегрувати в процесорі близько 30 транзисторів. Але зміна значень N_0 і y в виразі (1.1) дає змогу переміщувати точку відліку періоду прогнозування кількості транзисторів.

Кількість транзисторів у процесорах при одночасному зменшенні розмірів елементної бази потребує постійного впровадження нових технологій виробництва. Удосконалення технологічного процесу виробництва мікропроцесорів гарантувало безперервне збільшення кількості. Але сьогодні Intel підтвердила, що під час переходу на більш тонкі технологічні норми набагато складніше відповідати постулатам закону Мура. Отже, нерівномірність у розвитку технології виробництва

(рис. 1.3) привела до необхідності уточнення закону Мура (1.1). Аналіз динаміки збільшення кількості транзисторів у мікропроцесорах дав змогу узагальнити формулу (1.1) та сформувати такі варіанти закону [14]:

– швидкий закон Мура, де уу=2 з 1965 р. по 1985 р.;

– сучасний закон Мура, де уу≈1,6 з 1985 р. по 2005 р.;

– повільний закон Мура, де уу≈1,78 з 2005 р.

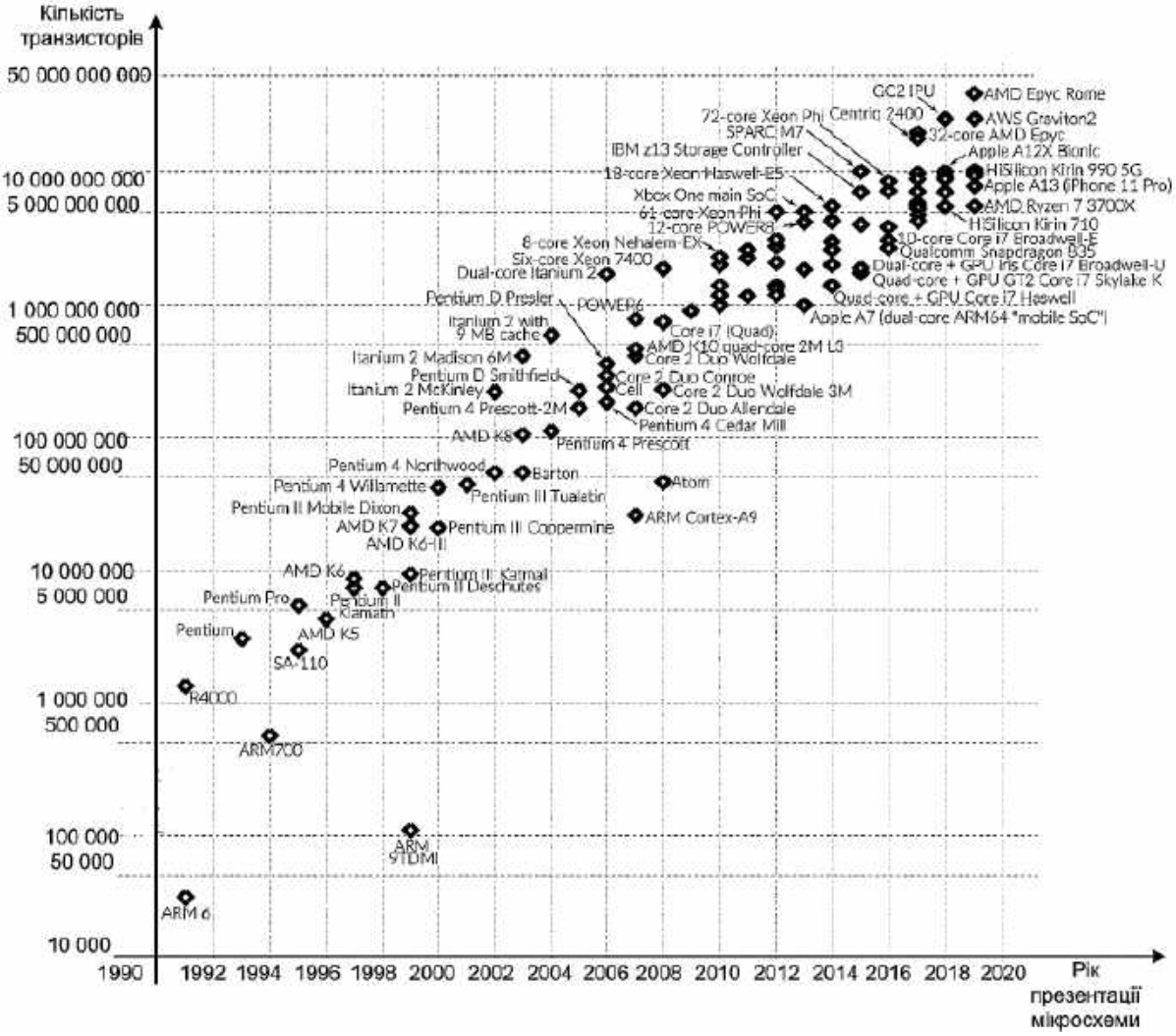


Рис. 1.3. Ілюстрація закону Мура: залежність кількості транзисторів у мікросхемі від року її випуску (у логарифмічній шкалі) (за даними ресурсу https://wikipedia.org/wiki/Transistor_count)

При цьому критичною точкою вважають останній доступний для практичного освоєння техпроцес у 7 нм. У 2019 р. глава компанії NVIDIA зазначив, що зараз закон Мура виражається у збільшенні кількості транзисторів лише на кілька відсотків щороку і подвоєння очікується лише кожні десять років [12, 13].

Приклад. Для певних моделей процесорів на кінець 2010 р. кількість транзисторів, які складають його ядро, досягла 75,8 млн. Їх розташовано на кристалі площею 189 мм^2. За законом Мура оцінити кількість транзисторів на одному кристалі процесора 2020 року випуску. Чи «працює» закон Мура, якщо відомо, що в 2020 р. у більш пізнішої версії

цього процесора ядро складають 12 млрд транзисторів, розташованих на площі 618 мм^2?

Кількість транзисторів, що розташовані на 1 мм^2 кристала процесора, у 2010 р. становить

$$\frac{75{,}8 \cdot 10^6}{189} = 0{,}4 \cdot 10^6 \text{ шт.}$$

За повільним законом Мура кількість транзисторів, розміщених на 1 мм^2 кристала процесора, подвоюється кожні 1,78 роки. За цих умов до 2020 р. ця кількість становитиме

$$N(2020) = N_{2010} 2^{\frac{y}{yy}} = 0{,}4 \cdot 10^6 \cdot 2^{\frac{(2020-2010)}{1{,}78}} \approx 19{,}7 \cdot 10^6 \text{ шт.}$$

За фактичними даними кількість транзисторів, що розташовані на 1 мм^2 кристала процесора 2020 р. випуску, досягла

$$\frac{12 \cdot 10^9}{618} \approx 19{,}42 \cdot 10^6 \text{ шт.}$$

Порівнюючи фактичні дані та дані, отримані за законом Мура, зазначимо, що вони майже збігаються. Отже, можна вважати, що закон Мура відбиває процеси розвитку комп'ютерної техніки та технологій.

Розвиток елементної бази комп'ютерів визначає темпи зростання швидкості передавання даних, збільшення щільності записів на носії, збільшення продуктивності та інших характеристик інформаційних систем. Тому тенденції розвитку за законом Мура, які спостерігаються для комп'ютерів, також характерні для показників цифрових інформаційних систем [14].

Приклад *За законами, аналогічними закону Мура, можна знайти такі характеристики інформаційних систем, як зміна вартості зберігання інформації або зміна пропускної здатності мережі та ін. Наприклад, закон Нільсена визначає, що швидкість підключення користувачів до мережі збільшується на 50 % щорічно або кожні 20 місяців [14].*

Хоча закон Мура виправдовує себе тривалий період часу, багато хто скептично ставиться до його дії в майбутньому. У 2007 р. сам Г. Мур заявив, що закон скоро перестане діяти через атомарну природу речовини, обмеження в швидкості світла та в зростанні продуктивності обчислювальних систем. Дійсно [13, 15]:

– зменшення електронних приладів не може відбуватися нескінченно, а розроблення більш малих транзисторів стає нерентабельним;

– максимальна тактова частота мікропроцесорів залежить від швидкості світла, яка має стале значення;

– зі збільшенням тактової частоти за законом Мура можна досягти квантової межі, значення якої обмежено сталою Планка;

– зменшення технологічних норм відносно розміру затвора

транзистора – довжини каналу, що пропускає або не пропускає крізь себе струм, – обмежено 7 нм для кремнію – основного напівпровідника при виробництві транзисторів;

– збільшення кількості транзисторів у процесорі обмежено технологічними нормами, що пов'язано з розкидом параметрів, який неможливо контролювати при пропорційному зменшенні розмірів;

– збільшення кількості транзисторів у процесорі обмежено допустимими значеннями енергоспоживання та тепловиділення, що призводить до необхідності впровадження нових технологій охолодження мікросхем;

Примітка *Чим більше транзисторів у процесорі, тим більший струм вони споживають, а отже, збільшується тепловиділення. Якщо не було б спаду в законі Мура, то вже після 2010 р. потужність, що виділяється транзисторами, досягла б такого значення, яке можна порівняти зі значенням тепловиділення на поверхні Сонця.*

– збільшення продуктивності внаслідок впровадження паралельності обчислювань обмежено законом Амдала: у випадку поділу завдання на частини, сумарний час його виконання в паралельній системі не може бути меншим за час виконання найповільнішої частини.

Приклад *Для програмного забезпечення існує варіант закону Мура, названий на честь засновника Microsoft Б. Гейтса. Цей закон стверджує, що швидкість програмного забезпечення зменшується на половину кожні півтора роки, що зводить нанівець усі переваги закону Мура. Серед основних причин цього називають: додавання надмірних непотрібних функцій, поганий код, небажання програмістів доопрацьовувати програми, неефективний менеджмент або часту зміну команди [14].*

Сучасний розвиток комп'ютерних технологій характеризується спадом продуктивності мікропроцесорів і уповільненням процесу зменшення елементної бази електроніки. Тривалість цього періоду суттєво залежить від упровадження нових наукових відкриттів. Так, фахівці пов’язують подальший розвиток комп'ютерних технологій з декількома напрямами [15, 16]:

1. Упровадження нових технологій для збільшення продуктивності процесорів, наприклад, перехід на нові технологічні норми, впровадження багатоядерних архітектур або тривимірних транзисторів.

Приклад *На початку 1960-х років почалось виробництво польових транзисторів на базі кремнію. Це дало змогу зменшити їхні розміри та енергоспоживання. З появою перших процесорів кількість транзисторів у них збільшувалася, а з цим і потужність обчислювальних систем. Зазначимо, що еволюція процесорів спрямована не лише на зменшення розмірів транзисторів, але й на змінення їхньої структури. Класичні, планарні або плоскі транзистори не використовують у процесорах з 2012 р. Їх замінили тривимірні транзистори.*

2. Упровадження нових матеріалів (порівняно з кремнієм), наприклад, антимоніду індію (InSb), арсеніду галію-індію (InGaAs), а можливо вуглецю,

як у формі нанотрубок, так і у формі графену, або випуск метал-повітряних транзисторів тощо.

3. Упровадження нових алгоритмів обчислення, наприклад реалізація паралельних обчислювань або створення квантових комп’ютерів.

Приклад *Початком квантової ері вважають 1900 р., коли М. Планк висунув гіпотезу про те, що енергія випускається і поглинається окремими квантами. Цю ідею підхопили та розвинули багато вчених того часу – Бор, Ейнштейн, Шредінгер та ін., що привело до створення та розвитку квантової фізики. Першим ідею квантових обчислень висловив у 1980 р. Юрій Манін, але заговорили про неї в 1981 р., коли Р. Фейнман на конференції з фізики обчислень зазначив, що неможна ефективно моделювати еволюцію квантової системи на класичному комп'ютері. Він запропонував елементарну модель квантового комп'ютера для проведення такого моделювання. На сьогодні квантові комп’ютері – це перспективний, дуже цікавий, але мало вивчений напрям комп’ютерних технологій.*

Тестові запитання та завдання для самоперевірки

1. Виберіть правильне визначення:

а) комп’ютерні технології – це сукупність документів і технологій з використанням засобів обчислювальної техніки та зв'язку, що реалізують інформаційні процеси;

б) комп’ютерні технології – широкий клас дисциплін, які належать до технологій створення, збереження, оброблення даних і керування ними із застосуванням обчислювальної техніки;

в) комп’ютерні технології – обчислювальні пристрої, які використовують для зберігання, відображення й оброблення даних.

2. Яке твердження є правильним:

а) реалізація функцій централізованого оброблення даних привела до широкого розповсюдження мейнфреймів у світі;

б) наявність машинної мови як основи взаємодії людини та мейнфрейму привела до широкого розповсюдження мейнфреймів у світі;

в) нативна підтримка десяткової арифметики привела до широкого використання мейнфреймів у найбільших фінансових установах світу?

3. Установіть відповідність «винахід – рік виходу» (наприклад, а – а, б – б та ін.):

Винахід	Рік виходу
а) персональний комп’ютер Altair	а) 1973
б) персональний комп’ютер PC IBM	б) 1977
в) комп’ютерна мережа ARCnet	в) 1971
г) опис комп’ютерної мережі Ethernet	г) 1981
д) мікропроцесор фірми Intel	д) 1975

4. Укажіть риси, характерні для ери PC & LAN:

а) мережі розраховано на необмежену кількість абонентів;

б) код програми-сервера дублюється програмами-клієнтами;

в) недотримується принцип відкритості систем;

г) широко впроваджують архітектуру «клієнт-сервер».

5. Виберіть характерні для WAN риси:

а) використання технології децентралізованого оброблення даних;

б) надійність і відмовостійкість як ключові показники мережі;

в) не дуже якісні канали зв'язку з відносно низькою швидкістю передавання даних;

г) орієнтовані на кінцеву кількість комп’ютерів, розташованих на фіксованій території.

6. Що таке закон Мура:

а) закономірність, яка описує взаємозв'язок швидкодії комп'ютера з його енергоспоживанням;

б) формула, що встановлює абсолютну межу збільшення обчислювальної потужності комп'ютера;

в) емпірична закономірність, за якою прогнозують збільшення кількості транзисторів у процесорах комп’ютерів;

г) доведена принципова неможливість створити ідеальний комп’ютер, який називають машиною Тьюринга?

7. Яке твердження є неправильним:

а) закон Амдала не дає змоги збільшувати тактову частоту процесорів відповідно до закону Мура;

б) технології зменшення розмірів процесорів стають нерентабельними;

в) розвиток елементної бази комп’ютерів визначає темпи збільшення багатьох характеристик інформаційних систем?

8. Що є перспективним напрямом подальшого розвитку комп’ютерних технологій:

а) повернення до перевіреної, добре апробованої технології застосування планарних транзисторів у процесорі;

б) збільшення площі ядра процесора для зменшення тепловиділення та енергоспоживання;

в) застосування кремнію як екологічного матеріалу для напівпровідникової техніки;

г) упровадження нових алгоритмів обчислення шляхом реалізації алгоритмів паралельних обчислень?

9. Серед заявлених характеристик процесора Athlon 64 від компанії AMD, який був представлений у квітні 2005 р., кількість транзисторів, що складають ядро процесора, досягла 76 млн. Їх розташовано на кристалі площею 84 мм^2. Для цього процесора за законом Мура оцініть можливу кількість транзисторів на одному кристалі в 2020 р.

Лекція 2. ЗАГАЛЬНІ ПРИНЦИПИ ПОБУДОВИ КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

План

1. Архітектура комп'ютерів. Загальні принципи організації комп'ютерних технологій.
2. Структура арифметико-логічного пристрою комп'ютера.
3. Модель OSI.

1. Архітектура комп'ютерів. Загальні принципи організації комп'ютерних технологій

Уже в 70-х роках XX ст. світові лідери з виробництва, аналізуючи досвід конкурентів та успіх продажів комп'ютерів, вибрали курс на створення їх єдиного модельного ряду. На ці висновки перш за все вплинули [17]:

– висока трудомісткість програмування та необхідність підтримання сумісності математичного забезпечення для комп'ютерів з різною будовою;

– наявні значні обмеження апаратних ресурсів;

– висока вартість апаратних і часових ресурсів, малий обсяг запам'ятовувального пристрою комп'ютерів тощо.

Саме тоді запроваджують поняття архітектури комп'ютера як концептуальної моделі, достатньої для розуміння користувачами, втіленої в певних компонентах, яка визначає принципи його проєктування, роботи та взаємодії один з одним і з оточенням. У той час на формування архітектури суттєво впливали:

– призначення комп'ютера: різну архітектуру було передбачено для військових, наукових або економічних розрахунків, керування та ін.;

– комерційна вигода, конкуренція та ринкова кон'юнктура;

– швидкість мініатюризації та нові винаходи, особливо ті, які спрямовані на збільшення обсягу пам'яті: саме досягнення певної межі обсягу пам'яті стимулювало змінення архітектури, методів програмування та організацію операційної системи.

Отже, основна ідея узагальненої архітектури – досягти для всіх елементів комп'ютерів однакових архітектурних ознак незалежно від їх призначення, а саме: загального набору команд, однакового способу адресації, стандартних інтерфейсів з'єднання з різноманітними периферійними пристроями тощо. Завдяки цьому вони виконували б ті ж самі програми з однаковими результатами для однакових вихідних даних [1, 17].

Основи теорії про архітектуру обчислювальних машин були закладені Джоном фон Нейманом ще в 1945 р. у статті «Попередній розгляд логічної конструкції електронно-обчислювального пристрою», де він висунув основні принципи будови арифметико-логічного пристрою комп'ютера та запропонував його структуру, фізично відокремивши процесорний модуль від пристрою зберігання програм та даних.

Примітка *Джон фон Нейман був генієм у багатьох областях. Коли він зацікавився обчислювальними машинами, його вважали одним із найзнаменитіших математиків світу. Працюючи в проєктах з розроблення комп'ютерів ENIAC і EDVAC, він сформував ідею комп'ютера з програмою, яку зберігають у пам'яті. Ця архітектура досі має назву «архітектура фон Неймана». Довгі роки її реалізовували в усіх комп'ютерах і мікропроцесорах, і навіть зараз більшість з принципів цієї архітектури залишаються застосовними.*

Структурно в архітектурі комп'ютера було виділено п'ять основних складових [18]: пам'ять; арифметико-логічний пристрій; пристрій керування; пристрої введення та виведення даних. Ця структура була реалізована в EDVAC – першій машині з програмою в пам'яті – і зараз є основою більшості сучасних цифрових комп'ютерів (рис. 2.1).

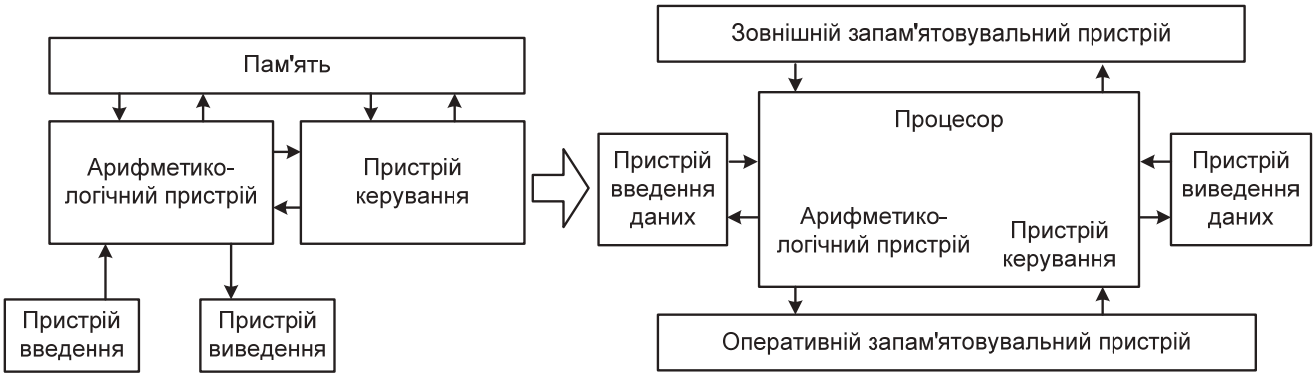


Рис. 2.1. Структурна схема архітектури фон Неймана та її розвиток у сучасних комп’ютерах

Архітектура фон Неймана стала покращанням результатів проєкту першого комп'ютера – ENIAC. Працюючи над ним, Дж. фон Нейман зазначав:

– створення комп'ютерів з великою кількістю перемикачів і кабелів потребує тривалого часу та великих витрат;

– програму слід подавати в пам'яті комп'ютера в цифровій формі разом з даними;

– десяткову арифметику, використану в ENIAC, слід замінити бінарною арифметикою.

Пізніш ці ідеї було узагальнено та відпрацьовано за роки розвитку комп’ютерів.

Основною функціональною частиною сучасного комп'ютера став процесор (рис. 2.1), який поєднав функції арифметико-логічного пристрою та пристрою керування. Він виконує основні операції з оброблення даних, керування роботою інших блоків, а також з перетворення інформації, що надходить з пам'яті та зовнішніх пристроїв.

Арифметико-логічний пристрій (англ. arithmetic and logic unit, ALU) – центральний елемент процесора, який за допомогою пристрою керування виконує найважливіші перетворення (починаючи з елементарних) даних,

зокрема:

– арифметичні операції – процедури оброблення даних, аргументи та результат яких є числами;

– логічні операції – операції над логічними виразами, які розглядають як умови для керування послідовністю виконання програми;

– операції над бітами – операції зсуву.

Пристрій керування забезпечує керування та контроль усіх пристроїв комп'ютера.

Запам'ятовувальні пристрої забезпечують зберігання вихідних і проміжних даних. При цьому в оперативному запам’ятовувальному пристрої зберігають інформацію, з якою комп'ютер працює безпосередньо в даний час (наприклад, резидентна частина операційної системи, прикладна програма, оброблювані дані); у зовнішньому запам’ятовувальному пристрої – інформацію великих обсягів для тривалого зберігання.

Під час роботи комп'ютера відбувається обмін інформацією між його складовими. Наприклад, операційна система зберігається у зовнішньому запам’ятовувальному пристрої (на жорсткому диску), але під час запуску резидентна частина операційної системи завантажується в оперативний запам’ятовувальний пристрій і зберігається там до завершення сеансу роботи.

Багато в чому будову комп’ютера визначають принципи створення, які пояснюють його архітектуру та конкретну систему команд. До цих принципів належать [17, 18]:

1. Принцип двійкового кодування, відповідно до якого всю інформацію в комп'ютері кодують бітами – двійковими цифрами 0 і 1.

2. Принцип однорідності пам'яті, згідно з яким команди та дані зберігаються в пам'яті комп’ютера та ззовні в ній не розрізняються. А отже, те ж саме значення в комірці пам'яті комп’ютера можна використовувати як дані, як команду або як адресу; розрізнити їх можна лише за способом звернення.

3. Принцип адресності, який передбачає, що структурно пам'ять комп'ютера складається з пронумерованих комірок, в які записують двійкові коди команд і даних.

4. Принцип програмного керування, який передбачає, що вирішення всіх завдань у комп'ютерах здійснюється програмним способом шляхом послідовного виконання в часі окремих команд, передбачених алгоритмом розв'язання задачі.

5. Принцип ієрархії, який означає, що комп'ютер будують як набір блоків, з'єднаних у структуру, які складаються з модулів (арифметичний пристрій, регістр, лічильник та ін.). Модулі будують з логічних елементів певного елементного базису тощо. Так само масив даних ієрархічно складається з окремих слів, які є послідовністю бітів. Складна програма складається з декількох програмних модулів, до складу яких входять підпрограми, подані у вигляді послідовності операторів, кожен з яких виконується як одна або декілька машинних команд, та ін.

Розглянемо деякі з цих принципів докладніше.

Уже зрозуміло, що створення комп’ютерів зумовлено необхідністю оброблення великої кількості числової інформації та проведення розрахунків. Унаслідок розвитку функцій комп’ютерів виникла необхідність в обробленні інформації інших типів (текстової, графічної, відеоінформації тощо). Тому, успадковуючи вже наявний досвід, цю інформацію також подавали в числовому вигляді. Але правила кодування – правила подання інформації в числах – для кожного її типу різні та визначаються відповідними стандартами. Наприклад, для кодування текстової інформації застосовують стандарти ASCII, Windows-1251, Unicode, для числової – стандарт IEEE 754 та ін., для графічної – RGB-модель тощо. При цьому спочатку ці стандарти було орієнтовано на людину, тобто вся інформація кодувалася в десятковій (інколи в шістнадцятковій) системі числення.

Принцип двійкового кодування, тобто перехід на двійкову арифметику, був обґрунтований математиками та конструкторами перших комп'ютерів у 50-х рр., коли підбирали систему числення для задоволення вимог інженерів і розробників програмного забезпечення, адже від її вибору залежали швидкість обчислень в комп'ютерах, місткість пам'яті, складність алгоритмів виконання арифметичних операцій.

Вибір двійкової системи зумовлено такими причинами:

1. Алгоритми виконання операцій у двійковій системі числення є простішими. Наприклад, таблиця множення в десятковій системі має розмірність 10х10 і містить 100 добутків усіх пар чисел, а в двійковій системі – лише чотири.

2. Двійкова система дає змогу застосовувати апарат булевої алгебри для виконання логічних перетворень.

3. Для фізичного подання чисел потрібні елементи, що здатні перебувати в одному з декількох стійких станів, кількість яких дорівнює основі прийнятої системи числення. Тому для зображення будь-якого числа в десятковій системі числення необхідні десять різних символів, тобто потрібні функціональні елементи, які мають десять стійких станів. Створення електронних елементів з великою кількістю стійких станів суттєво ускладнено, але технічно просто реалізувати пристрій з двома станами (наприклад, є струм – немає струму, світло – темно, увімкнено – вимкнено тощо).

4. Двійкова система забезпечує максимальну стійкість процесу передавання даних як між окремими складовими комп’ютера, так і між пристроями, розташованими на великій відстані.

Таким чином, в усіх комп’ютерах інформацію незалежно від типу кодують двійковими цифрами – бітами, утворюючи двійкову послідовність. Але залежно від типу правила запису бітів у послідовність визначає відповідний формат.

Згідно з принципом адресності основну пам'ять комп'ютера структурно складають пронумеровані комірки, які в довільний момент часу

доступні процесору. Для доступу до вмісту будь-якої комірки слід скористатися її номером – адресою.

Для збереження у відповідних комірках пам'яті двійкових послідовностей їх поділяють на одиниці інформації (слова).

Машинне слово – послідовність двійкових розрядів фіксованої довжини для збереження у комірці, що сприймається як єдине ціле й обробляється за допомогою набору команд або апаратного забезпечення процесора. Кількість розрядів у машинному слові називають його розмірністю, або довжиною [19].

Машинне слово – базове поняття комп'ютерних технологій, його розмірність визначає такі характеристики апаратної складової комп'ютера:

– кількість даних, яку процесор може обробити за одну операцію;

– розрядність шини даних;

– найбільша частина даних, яку за одну операцію можна передати в робочу пам'ять та з неї;

– максимальне значення беззнакового цілого числа, яке підтримує процесор, тощо.

Також розрізняють поняття стандартного машинного слова – машинне слово, розмірність якого збігається з розрядністю процесора.

Примітка *Стандартне машинне слово – машинозалежна та платформозалежна величина. На ранніх етапах розвитку комп'ютерних технологій модулі пам'яті коштували дорого, тому вибір розмірності слова впливав на точність обчислювань і вартість комп'ютерів. У науково-технічних комп'ютерах 48-бітове машинне слово було найпопулярнішим. Воно давало змогу виразити дійсне число з 10 знаками після точки. У 50 – 60-х рр. у багатьох комп'ютерах, що вироблялися в США, розмірність машинного слова була кратна шести бітам, тому що використовували шестибітове кодування символів англійського алфавіту. У сучасних комп'ютерах довжина слова є числом, кратним степеню двійки; при цьому використовуються восьмибітові символи.*

Більшість команд процесора використовують стандартне машинне слово для оброблення даних, розглядаючи його як мінімально адресовану комірку пам'яті, а отже, на цей розмір суттєво впливає архітектура процесора. Залежність розміру стандартних машинних слів від архітектури подано в табл. 2.1.

Зазвичай як машинне слово розуміють 16 бітів, але цей термін слід розглядати в контексті. Якщо про «слово» говорять програмісти на C/C++, то це 16 бітів, якщо на асемблері, то це 32 біти, розмірність якого збігається з розрядністю процесора, який програмують, тощо.

Розмірність слова закладена в основу роботи багатьох протоколів, наприклад, типовий заголовок IP-пакета складається з п'яти 32-бітових слів, довжина IP-адреси дорівнює 32 бітам тощо. Крім того, використовують укорочене машинне слово (у мовах програмування йому відповідає термін short) і подвійне машинне слово (термін long).

Таблиця 2.1

Розмір машинного слова залежно від архітектури процесора [19]

Рік	Архітектура	Розмір слова, w бітів	Допустимий розмір цілих чисел
1952	IBM 701	36	$\frac{1}{2}$w, w
1960	CDC 1604	48	w
1965	PDP-8	12	w
1968	БЭСМ-6	48	w
1970	PDP-11	16	$\frac{1}{2}$w, w
1971	Intel 4004	4	w
1978	Intel 8086	16	$\frac{1}{2}$w, w
1989	Intel 80486	16	$\frac{1}{2}$w, w, 2w, 4w
2000	Itanium	64	8 bit, $\frac{1}{4}$w, $\frac{1}{2}$w, w
2002	XScale	32	w
2010	RISC-V	32	$\frac{1}{4}$w, $\frac{1}{2}$w, w, 2 w, 4 w

Кожне машинне слово характеризується своєю інформаційною місткістю (обсягом).

Інформаційна місткість визначає кількість можливих поєднань, що отримують для n-розрядного слова.

Кожен двійковий розряд може набувати двох значень, тобто його інформаційна місткість дорівнює двом, два розряди – чотирьох значень, тому інформаційна місткість двох розрядів дорівнює чотирьом. У загальному випадку інформаційну місткість n-розрядного слова оцінюють за формулою

$$V = 2^n, \qquad (2.1)$$

де n – кількість розрядів машинного слова.

Приклад. Відповідно до ISO 8895-1 сучасна версія стандарту ASCII використовує один байт для кодування одного символу текстової інформації. Знайдіть інформаційну місткість стандарту для визначення максимально можливої кількості кодів для кодування символів за допомогою ASCII.

Пам'ятаємо, що 1 байт дорівнює 8 бітам. Застосовуючи формулу (2.1), отримаємо

$$V = 2^n = 2^8 = 256.$$

Отже, при кодуванні символів текстової інформації двійковою послідовністю розміром 8 бітів стандарт ASCII має інформаційну місткість 256, що дає змогу закодувати 256 символів.

2. Структура арифметико-логічного пристрою комп’ютера

Проілюструємо, як розглянуті вище принципи відобразилися на апаратній частині комп'ютера і реалізовані в структурі арифметико-логічного пристрою. Нагадаємо, що саме цей пристрій гарантує, що

комп'ютер здатний виконувати базові математичні операції.

Залежно від функцій у структурі пристрою виділяють дві частини [18]:

– мікропрограмний пристрій, або пристрій керування, який задає послідовність мікрокоманд (команд);

– операційний пристрій, який реалізує задану послідовність мікрокоманд (команд).

Ці частини реалізовані як швидкодійні електронні логічні схеми високої щільності, кожна з яких містить сотні тисяч елементів.

Операційна частина арифметико-логічного пристрою (рис. 2.2) умовно складається з набору регістрів Pr1 – Pr7, призначених для оброблення інформації N_1, …, N_s, яка надходить з оперативної (пасивної) пам’яті комп’ютера.

Рис. 2.2. Спрощена структурна схема арифметико-логічного пристрою

Регістр – логічний пристрій для фізичної реалізації комірки пам’яті всередині арифметико-логічного пристрою, яку використовують для збереження n-розрядних двійкових чисел (операндів) і їх перетворення. Для забезпечення зв'язків між регістрами на вході кожного з них зібрано відповідні логічні схеми для оброблення вхідних даних згідно з мікрокомандами, що надходять до операційного пристрою з пристрою керування.

Типовими регістрами, що входять до складу пристрою, є [17, 18]:

– Pr1 – суматор (акумулятор) – головний регістр арифметико-логічного пристрою, де формується результат обчислень, використовується також для зберігання проміжних результатів арифметичних і логічних операцій та інструкцій введення-виведення;

– Pr2, Pr3 – регістри загального призначення, основною функцією яких є збереження доданків, множників, діленого або дільника (залежно від виконуваної операції) і проміжних результатів обчислень, необхідних

для роботи процесора;

– Pr4 – адресний регістр, призначений для запам'ятовування (іноді для формування) адреси операндів і результату;

– Pr5 – регістри прапорів, призначені для збереження ознак результатів операцій, що виконуються;

– Pr6 – k індексних регістрів, вміст яких використовують для формування адрес;

– Pr7 – і допоміжних регістрів, які за бажанням можна використовувати як акумулятори, індексні регістри або регістри загального призначення при запам'ятовуванні проміжних результатів.

Частина операційних регістрів арифметико-логічного пристрою є програмно-доступними, тобто вони можуть бути адресовані в команді для виконання операцій над своїм вмістом. До таких регістрів належать: суматор (регістр Pr1), індексні регістри Pr6 і деякі допоміжні регістри. Інші регістри не можна адресувати в програмі, тобто вони є програмно-недоступними [18].

Результати обчислень передаються до оперативного запам’ятовувального пристрою по кодових шинах Y_1, …, Y_s для запису.

Сучасні комп'ютери автоматично реалізують кілька сотень різних команд, але їх виконання базується на використанні найпростіших мікрооперацій типу додавання (тому Pr1 називають суматором) і зсуву. Це дає змогу мати єдиний арифметико-логічний пристрій для виконання будь-яких команд, пов'язаних з переробленням даних. Блок керування генерує відповідну послідовність сигналів, що ініціюють у пристрої виконання певних мікрооперацій згідно з їх кодом і сигналами сповіщення. При цьому розрізняють (див. рис. 2.2) [17, 18]:

– зовнішні мікрокоманди – мікрокоманди A_1, A_2, ..., A_n, які надходять до арифметико-логічного пристрою ззовні та приводять до перетворення даних N_1, …, N_s;

– внутрішні мікрокоманди – мікрокоманди P_1, P_2, …, P_m, що генерує арифметико-логічний пристрій при виконанні мікропрограми, які впливають на пристрій керування, змінюючи природний порядок виконання мікрокоманд.

Приклад *Залежно від результату обчислень арифметико-логічний пристрій може згенерувати такі ознаки: ознаку переповнення, ознаку від'ємного числа, ознаку рівності нулю всіх розрядів числа та ін.*

Складність логічної структури залежить від кількості різних мікрооперацій, необхідних для виконання всього комплексу завдань, поставлених перед арифметико-логічним пристроєм. Усі операції, що виконуються в цьому пристрої, є логічними операціями (функціями), які можна поділити на такі групи [20]:

– операції двійкової арифметики для чисел з фіксованою та рухомою точками;

– операції десяткової арифметики;

– операції індексної арифметики (при модифікації адресів команд);

– операції спеціальної арифметики;

– логічні операції;
– операції над алфавітно-цифровими полями.

Приклад *До арифметичних операцій належать операції додавання, віднімання, віднімання модулів, множення та ділення. Групу логічних операцій становлять операції АБО (диз'юнкції), І (кон'юнкції), виключного АБО. Спеціальні арифметичні операції реалізують нормалізацію, арифметичний і логічний зсуви. Існує велика група операцій редагування алфавітно-цифрової інформації тощо.*

Отже, сучасний арифметико-логічний пристрій оперує чотирма типами інформаційних об'єктів: логічними, цифровими, байтовими й адресними, автоматично виконуючи кілька сот різних мікрокоманд.

Приклад *При виконанні команди передавання керування за умовою під час прийняття рішення про необхідність переходу в програмі арифметико-логічного пристрою за результатами порівняння:*
- тричі збільшується на одиницю лічильник команд;
- двічі виконується читання з внутрішньої пам'яті даних;
- виконується арифметичне порівняння двох змінних;
- формується адреса переходу.
Ці операції арифметико-логічний пристрій виконує менш ніж за 2 мкс.

Під час виконання завдання на вхід арифметико-логічного пристрою надходять [20, 21]:

– вхідні дані – операнди – n-розрядні двійкові числа, що беруть участь у арифметичних і логічних перетвореннях (розрядність операнда збігається з розрядністю машинного слова);

– прапор (ознака) – ознака, отримана під час виконання попередньої операції, наприклад біт переносу;

– команда, необхідна для виконання завдання, подана відповідним кодом операції.

На виході отримують результат відповідної операції та можливий набір прапорів (ознак), які враховують у подальших завданнях (рис. 2.3).

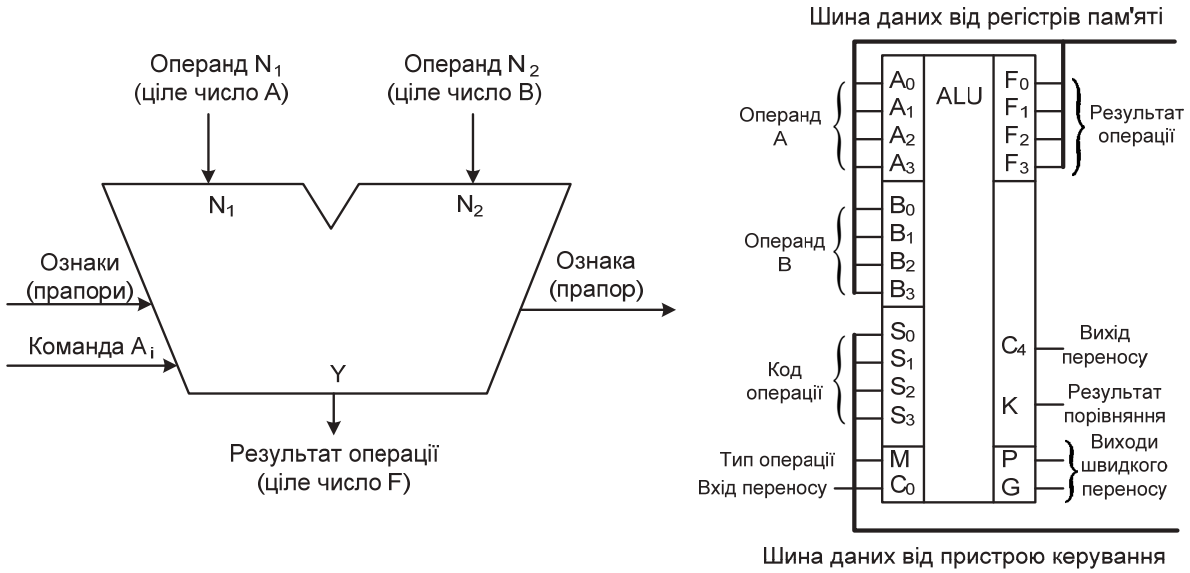


Рис. 2.3. Символьне подання та умовне графічне позначення 4-бітового арифметично-логічного пристрою моделі 74181

Команда машинної програми (машинна команда або мікрокоманда) – елементарна інструкція, яку комп’ютер виконує автоматично без додаткових указівок і пояснень.

Команди зберігаються в комірках (регістрах) пам'яті комп'ютера і виконуються в природній послідовності залежно від їх положення в програмі. Адресу (номер) чергового регістру пам'яті, з якого вилучають наступну команду, вказує спеціальний пристрій – лічильник команд у пристрої керування арифметико-логічного пристрою. За необхідності за допомогою спеціальних команд послідовність команд можна змінювати: або на підставі аналізу результатів виконаних обчислень під впливом внутрішніх мікрокоманд P_1, P_2, …, P_m, або безумовно.

У машинній команді виділяють дві частини [20]:

– операційну – група розрядів команди, призначених для подання коду операції для виконання комп’ютером;

– адресну – група розрядів команди, до яких записують коди адреси (адрес) регістрів пам'яті комп’ютера, призначених для оперативного збереження операндів, залучених для виконання команди.

У загальному випадку за кількістю записуваних адрес команди можна поділити на безадресні, одно-, дво- і триадресні (рис. 2.4) [21].

Безадресні команди використовують разом з командами іншої адресності. Вони містять лише код операції, припускаючи, що інформація для неї заздалегідь поміщена у відповідні регістри.

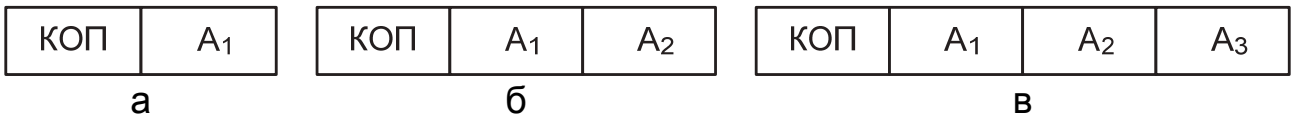


Рис. 2.4. Типова структура машинних команд

Коди операцій залежать від апаратної та програмної складових комп’ютера. Наприклад, для 4-бітового арифметично-логічного пристрою моделі 74181 (див. рис. 2.3) усі можливі коди операцій наведено в табл. 2.2.

На рис. 2.3 видно, що код операції визначають входи S_0 … S_3, а також вхід M, значення якого залежить від типу операції: при M = 1 арифметико-логічний пристрій виконує логічні операції (згідно з табл. 2.2), при M = 0 – арифметичні [20].

Типову структуру одноадресної команди показано на рис. 2.4, а, де КОП – код операції; A_1 залежно від модифікації команди може позначати або адресу регістра, де зберігається операнд, або адресу регістра, куди слід помістити результат операції. Прикладами одноадресної команди є команда інверсії бітів операнда або бітовий зсув [21].

Структуру двоадресної команди подано на рис. 2.4, б, де A_1 – зазвичай адреса регістра, де зберігають перший операнд і куди після завершення операції записують її результат; A_2 – адреса регістра, де зберігають другий операнд.

Таблиця 2.2

Взаємозв'язок між кодом і змістом операції для арифметико-логічного пристрою моделі 74181

Сигнали на входах S				Сигнали на вході M	
S_3	S_2	S_1	S_0	M=1 (логічні операції)	M=0 (арифметичні операції)
0	0	0	0	$\overline{A}$	$A+C_0$
0	0	0	1	$\overline{A \vee B}$	$(A \vee B)+C_0$
0	0	1	0	$\overline{A}B$	$(A \vee \overline{B})+C_0$
0	0	1	1	0000	$1111+C_0$
0	1	0	0	$\overline{AB}$	$A+\overline{AB}+C_0$
0	1	0	1	$\overline{B}$	$(A \vee B)+A\overline{B}+C_0$
0	1	1	0	$A \oplus B$	$A+\overline{B}+C_0$
0	1	1	1	$A\overline{B}$	$A\overline{B}+1+C_0$
1	0	0	0	$\overline{A} \vee B$	$A+AB+C_0$
1	0	0	1	$\overline{A \oplus B}$	$A+B+C_0$
1	0	1	0	B	$A \vee \overline{B}+AB+C_0$
1	0	1	1	AB	$1111+AB+C_0$
1	1	0	0	1111	$A+A+C_0$
1	1	0	1	$A \vee \overline{B}$	$A \vee B+A+C_0$
1	1	1	0	$A \vee B$	$A \vee \overline{B}+A+C_0$
1	1	1	1	A	$1111+A+C_0$

Типова структура триадресної команди містить адреси трьох регістрів (рис. 2.4, в). При цьому A_1 і A_2 – адреси регістрів, де розміщено операнди, що беруть участь у операції; A_3 – адреса регістра, куди після завершення операції поміщують її результат.

Приклад. На вхід арифметично-логічного пристрою моделі 74181 надійшла машинна команда, подана мовою символічного кодування, вигляду

1001 0	0010	0101

.

Розшифруйте цю команду.

Вихідна операція двоадресна. Відповідно до табл. 2.2 двійковий код операції 1001 при M=0 відповідає арифметичній операції додавання типу $A+B+C_0$. Адреса регістра, де зберігають операнд A, – 0010. Адреса регістра, де зберігають операнд B, – 0101. Після додавання результат записують у регістр з адресою 0010.

У кодах машини всі команди містять тільки двійкові цифри. Для кожної складової комп'ютера та ГІС-додатка залежно від особливостей застосування машинних команд визначено правила їх побудови, ураховано особливості передавання даних, рівень, на якому їх використовують, тощо. Наприклад, у протоколі обміну даними Modbus RTU

кількість операцій визначена специфікою застосування, а коди операції пов’язані з регістром, де зберігають / куди записують операнди (табл. 2.3) [22].

Таблиця 2.3

Коди операцій, їх опис і прив'язка до вхідних / вихідних регістрів пристрою

Код	Опис	Діапазон адрес регістрів	Код	Опис	Діапазон адрес регістрів
1 (0x01)	Зчитування даних з регістра прапорів (Coil)	1 – 9999	5 (0x05)	Запис одного значення в регістр прапорів	1 – 9999
2 (0x02)	Зчитування даних з регістра вхідних даних (Discrete Input)	10001 – 19999	6 (0x06)	Запис одного значення в регістр зберігання	40001 – 49999
3 (0x03)	Зчитування значень з декількох регістрів зберігання (Holding Registers)	40001 – 49999	15 (0x0F)	Запис декількох значень в регістри прапорів (Coils)	1 – 9999
4 (0x04)	Зчитування значень з декількох вхідних регістрів (Input Registers)	30001 – 39999	16 (0x10)	Запис декількох значень в регістри зберігання	40001 – 49999

Примітка. Протокол використовує шістнадцяткову систему числення під час обміну даними між пристроями.

Протокол Modbus RTU оснований на архітектурі ведучий (master)–ведений (slave) (у сучасній версії стандарту Modbus – client-server), регламентує роботу контролерів у мережі, що спілкуються по шині Modbus шляхом реалізації послідовності операцій, які складаються із запиту та відповіді. При цьому в структурі запиту можна виділити такі елементи:

– адреса slave-пристрою, до якого формують запит;

– код операції;

– початкова адреса регістра, з якого слід зчитати дані;

– кількість регістрів, що опитують;

– контрольна сума для забезпечення достовірності обміну даними.

Приклад. Сформувати запит master-пристрою для Modbus RTU на читання даних з регістрів з адресами 0x0200, 0x0201 і 0x0202, які зберігаються в slave-пристрої з адресою 0x01.

За умовами прикладу реалізується запит з кодом операції 0X03 – зчитування даних з декількох регістрів зберігання, для опитування трьох регістрів slave-пристрою. З урахуванням вихідних даних структуру та двійковий код запиту зображено на рис. 2.5, а, де значення контрольної суми вказано довільно (принципи її розрахунку буде розглянуто пізніше).

На рис. 2.5, б наведено приклад структури відповіді на запит slave-пристрою. У відповіді різняться декілька байтів (позначено кольором): вказано, скільки байтів зчитано з регістрів і передано до master-пристрою, а також ці дані. Також з урахуванням того, що передаються інші дані, змінюється значення контрольної суми. Отже (відповідно до рис. 2.5, б), зі slave-пристрою зчитано й отримано такі значення:

– з регістра з адресою 0x0200 – 0x00B1;

– з регістра з адресою 0x0201 – 0x1F40;

– з регістра з адресою 0x0202 – 0x513D.

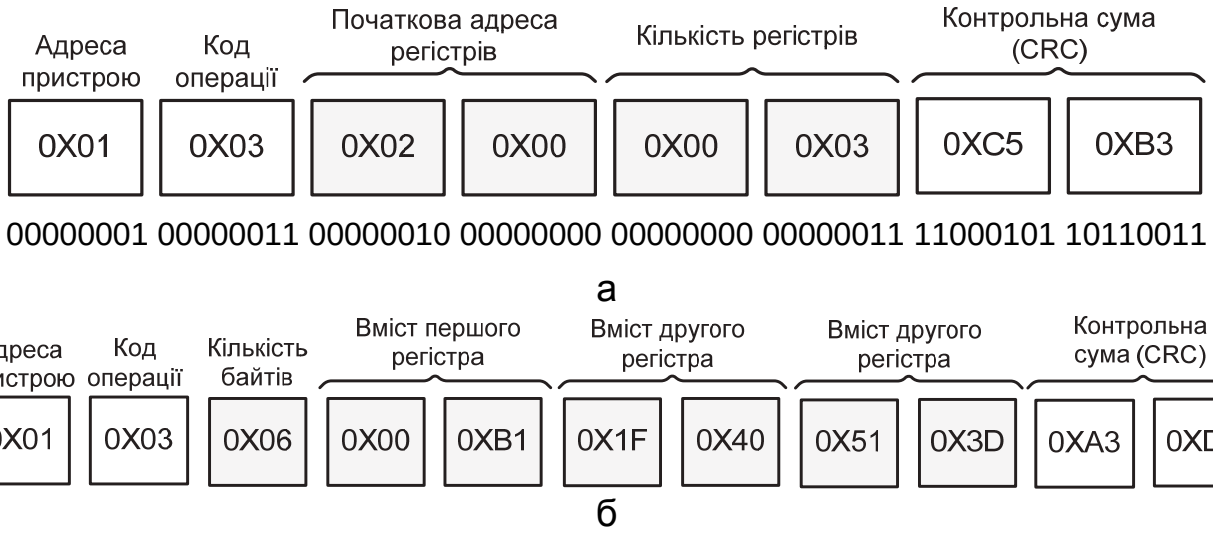


Рис. 2.5. Приклади машинних команд протоколу Modbus RTU:
а – структура та двійковий код запиту; б – структура відповіді

Комп’ютерні технології, їх функції ускладнюються, але користувачі цього не помічають через те, що спрощуються інтерфейси їх взаємодії. Комп'ютери й інформаційні системи стають все більш дружніми та зрозумілими для людини, яка не є фахівцем у галузі комп’ютерних технологій. Це стало можливим насамперед тому, що користувачі та їх програми взаємодіють з комп'ютерами за допомогою спеціального (системного) програмного забезпечення через операційну систему.

Операційна система – системне програмне забезпечення, яке поєднує комплекс взаємозв'язаних програм, призначених для керування ресурсами комп'ютера й організації взаємодії з користувачем, і використовує інші програми комп’ютера. Вона займає місце між апаратним забезпеченням комп’ютера (hardware) з його мікроархітектурою, машинною мовою та вбудованими мікропрограмами (драйверами) і прикладними програмами і є інтерфейсом, що ізолює апаратну частину комп'ютера від прикладних програм користувачів [23].

Примітка *Попередниками операційних систем вважають службові програми (завантажувачі) і бібліотеки підпрограм, що найчастіше використовувалися, які почали розробляти для комп'ютерів наприкінці 1940-х рр. Службові програми мінімізували фізичні маніпуляції оператора з обладнанням, а бібліотеки давали змогу уникнути багаторазового програмування однакових дій (здійснення операцій введення-виведення, обчислення математичних функцій тощо).*

Кроку впровадження операційної системи передували еволюція розвитку комп’ютерних технологій і реалізація кількох ідей, зокрема:

1. Пакетний режим – спосіб оптимізації використання дорогих обчислювальних ресурсів та уникнення можливого простою процесору шляхом формування черги на виконання програм, довжина якої корегується самостійним завантаженням системою програм в оперативну пам'ять із зовнішніх носіїв.

2. Поділ часу – спосіб розподілу обчислювальних ресурсів між

багатьма користувачами шляхом багатозадачності, коли частина завдань (наприклад, введення або редагування даних оператором) виконується в діалоговому режимі, а частина (наприклад, потужні обчислення) – в пакетному. Це сприяло розробленню нових інтерактивних програм і дало змогу багатьом користувачам одночасно взаємодіяти з одним комп'ютером, знижуючи вартість надання обчислювальних потужностей.

3. Поділ повноважень – спосіб блокування можливості зміни програми, що виконується, або даних певної програми в пам'яті комп'ютера іншою програмою (навмисно або помилково), а також зміни самої системи прикладною програмою. Розвиток цієї ідеї в операційних системах відбився в архітектурі процесора, що реалізовувала два режими його роботи – реальний (коли програмі, що виконується, доступний весь адресний простір комп'ютера) і захищений (коли доступність адресного простору обмежена діапазоном, виділеним під час запуску програми на виконання).

4. Масштаб реального часу – спосіб синхронізації виконання програм із зовнішніми фізичними процесами шляхом створення спеціалізованої операційної системи для надання необхідного та достатнього набору функцій проєктування, розроблення та функціонування систем реального часу на конкретному апаратному обладнанні.

5. Файлова система – спосіб зберігання даних у доступних і зручних для використання зовнішніх запам’ятовувальних пристроях.

Таким чином, операційна система виконує такі найважливіші функції:

1. Відповідає за ефективний розподіл обчислювальних ресурсів та організацію надійних обчислень.

2. Реалізує взаємодію користувача з комп'ютером і програмами шляхом введення команд операційної системи або вибору можливих дій, запропонованих операційною системою.

3. Здійснює доступ програм до апаратної складової комп'ютера у випадках, коли службові програми та програми користувачів запитують дозвіл на виконання тих операцій, що досить часто використовуються в будь-якій програмі.

Приклад *Розробникам програмного забезпечення операційна система дає змогу абстрагуватися від деталей реалізації та функціонування пристроїв через мінімально необхідний набір функцій (реалізація введення-виведення, запуск або зупинення будь-якої програми, отримання додаткового регістра пам'яті тощо). Розробники звертаються до програмної підсистеми з відповідним запитом та отримують від неї необхідні функції та послуги.*

Системні функції визначають можливості, які операційна система дає додаткам, що виконуються під її керуванням. Такі системні запити або явно прописують програмісти в тексті програми, або підставляються автоматично системою програмування на етапі трансляції вихідного тексту програми, яку розробляють. Системні запити також реалізують у вигляді команд.

3. Модель OSI

Широке впровадження комп'ютерних технологій та еволюція розвитку комп'ютерів (див. рис. 1.1) привели до об'єднання багатьох обчислювальних пристроїв та інформаційних систем, реалізованих на їх основі, в єдині мережні системи. Збільшення кількості пристроїв, що поєднувалися, призвело [11]:

– до суттєвого ускладнення організації фізичного інтерфейсу між пристроями, що об'єднуються в мережу;

– постійного узгодження програмно-апаратного забезпечення;

– необхідності перенавчання кадрів.

Суть сучасних комп'ютерних технологій – це поєднання різного обладнання (з огляду на організацію обчислювального процесу, архітектуру, систему команд, розрядність процесора, шини даних тощо) для вирішення різноманітних завдань оброблення даних. Це потребує створення фізичних інтерфейсів, які забезпечують взаємну сумісність під'єднання пристроїв.

Приклад *Багато користувачів стикалися із випадками несумісності обчислювальних, телекомунікаційних та інформаційних пристроїв і програм, зокрема:*
- коли програмне забезпечення, яке добре працює на одному комп'ютері, не працює на іншому;
- системні блоки одного обчислювального пристрою не стикуються з апаратною частиною іншого;
- інформаційна система університету не бажає обробляти дані, підготовлені для цієї системи вдома.

Отже, від успішного вирішення проблеми сумісності програмних та апаратних засобів залежить прогрес сучасних комп'ютерних технологій. Тому всі виробники дотримуються загальноприйнятих правил побудови обладнання, а розвиток комп'ютерної галузі відображається у відповідних стандартах: будь-яка нова технологія набуде «законного» статусу лише за умови, що її зміст закріплено у відповідному стандарті [8, 11].

Основою стандартизації сучасних комп'ютерних технологій став принцип ієрархії та реалізований на його основі багаторівневий підхід до розроблення пристроїв мережної взаємодії.

Принцип ієрархії означає, що складну обчислювальну функцію $F = (F_1, F_2, ..., F_i, ...)$ для реалізації обчислювачем, який складається з блоків $S = (S_1, S_2, ..., S_i, ...)$, подають композицією функцій F_i – декомпозують (розбивають на декілька простих функцій) – таким чином, щоб кожну функцію F_i виконував певний блок S_i [17]. Для кожного блока S_i чітко визначають функцію F_i і правила взаємодії (інтерфейси), що дає змогу спрощувати завдання при їх виконанні та навіть модифікувати окремі блоки без зміни решти системи. Декомпозиція дає змогу застосувати багаторівневий підхід, згідно з яким (рис. 2.6) [11]:

– усю множину блоків, що вирішують окремі завдання, розбивають на

групи;

– групи впорядковують за рівнями, утворюючи ієрархію так, щоб усі блоки групи для виконання своїх завдань зверталися із запитами тільки до блоків сусіднього нижнього рівня, а результати роботи – передавали сусідньому вищому рівню;

– для кожного проміжного рівня вказують сусідні рівні, які безпосередньо прилягають до нього і розташовані вище та нижче.

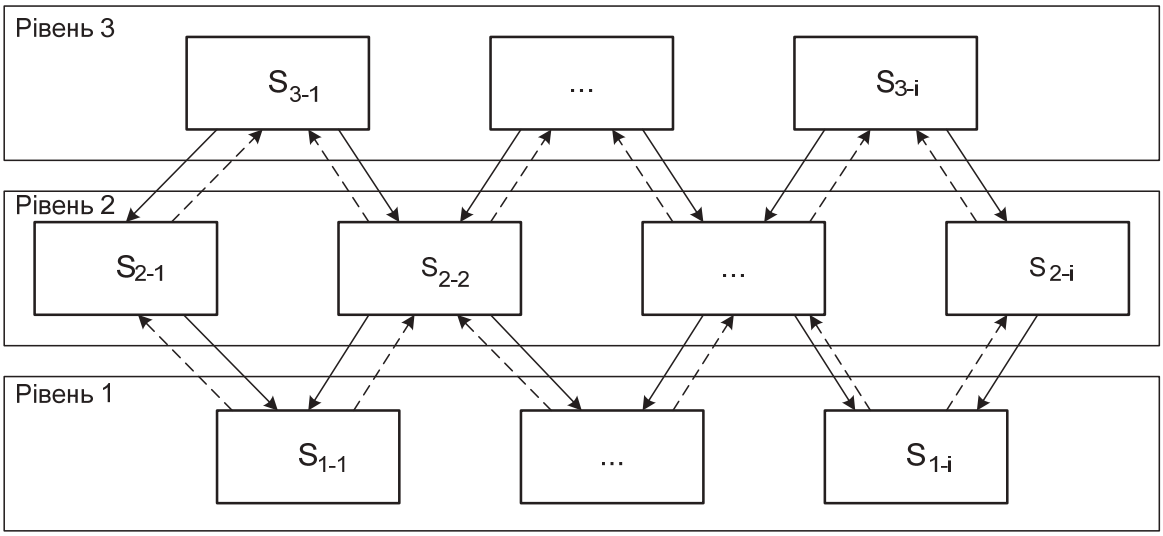


Рис. 2.6. Багаторівневий підхід взаємодії пристроїв у мережі

Переваги багаторівневого підходу:

– досягається відносна незалежність кожного з рівнів;

- з'являється можливість автономного розроблення рівнів та їх модифікації;

– відповідно до принципу ієрархії забезпечується можливість поділу завдань, що вирішують.

Приклад *Блоки нижнього рівня вирішують завдання, пов'язані з маршрутизацією даних при їх передаванні, а блоки верхнього рівня забезпечують надійність передавання даних між відправником і одержувачем у межах мережі.*

Багаторівневий підхід мережної взаємодії пристроїв передбачає, що в процесі обміну даними беруть участь дві сторони. Тому учасники обміну повинні ухвалити низку угод, наприклад, щодо рівнів і форми електричних сигналів, способів визначення розміру повідомлень для передавання даних, методів контролю та забезпечення достовірності передавання даних тощо. Розрізняють два основних типи подібних угод [8, 11]:

1. Протокол – формалізовані правила, що визначають послідовність і формат повідомлень, якими обмінюються компоненти комп’ютерних технологій на одному рівні, але в різних комп'ютерах.

2. Інтерфейс – формалізовані правила, які регламентують процес реалізації запитів блоків сусідніх рівнів одного комп’ютера, що визначають набір сервісів, які надаються цим рівнем сусідньому рівню.

У загальному випадку протоколи визначають правила взаємодії блоків одного рівня різних комп’ютерів, а інтерфейси – правила взаємодії блоків сусідніх рівнів одного комп’ютера (рис. 2.7). Отже, блоки кожного рівня обробляють правила власного протоколу та інтерфейси сусіднього рівня. При цьому зв'язок між поняттями «протокол» та «інтерфейс» не завжди однозначний: інтерфейс може поєднувати елементи протоколу, а протокол – охоплювати кілька інтерфейсів. Але основна ідея використання стандартних інтерфейсів і протоколів – уніфікація між- та внутрішньосистемних і між- та внутрішньомережних зв'язків для підвищення ефективності комп’ютерних систем [8].

Ієрархічно організований набір протоколів, достатній для організації взаємодії комп’ютерів у мережі, називають стеком комунікаційних протоколів.

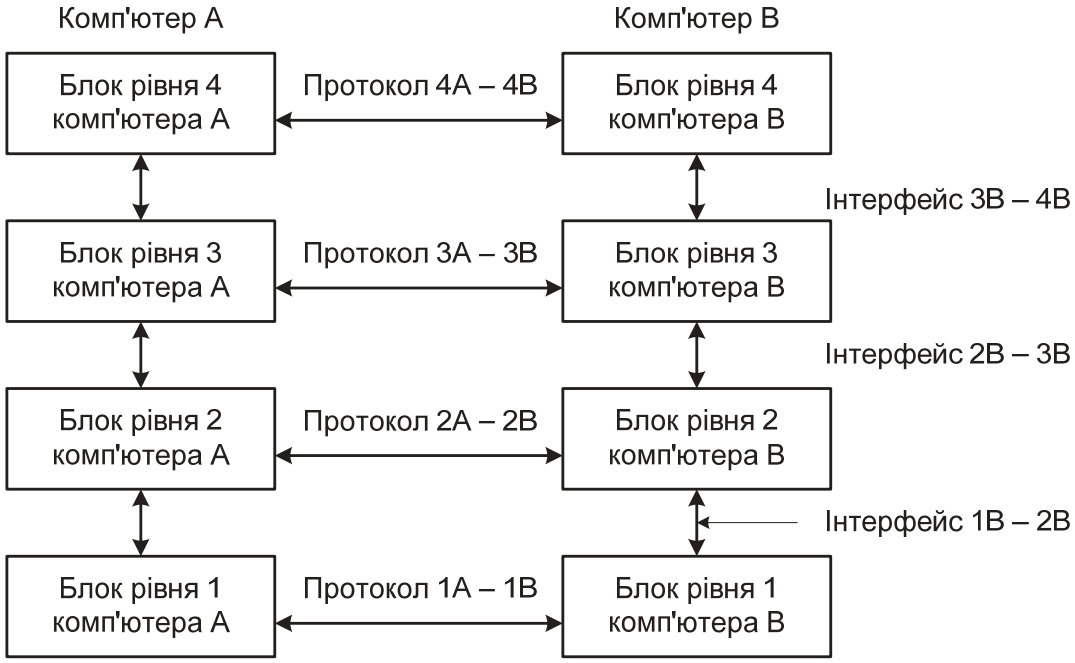


Рис. 2.7. Мережна взаємодія двох комп’ютерів

Комунікаційні протоколи можуть бути реалізовані програмно і апаратно. При цьому протоколи нижніх рівнів часто реалізуються комбінацією програмних і апаратних засобів, а протоколи верхніх рівнів – програмними засобами.

З огляду на те, що сучасні комп’ютерні технології – це комплексна робота комп'ютерів, комунікаційного обладнання, операційних систем і мережних додатків, протоколи реалізуються не тільки комп'ютерами, а й іншими мережними пристроями (концентраторами, мостами, комутаторами, маршрутизаторами тощо). Залежно від типу пристрою у ньому повинні бути вбудовані засоби для реалізації певного набору протоколів.

Базовим поняттям будь-яких міжнародних стандартів та угод у сфері застосування інформаційних технологій та розроблення інформаційних

систем є поняття відкритої системи (Open system).

Відкрита система – вичерпний та узгоджений набір міжнародних стандартів на інформаційні технології, складений згідно з відкритими специфікаціями, які визначають інтерфейси, служби та формати, що їх підтримують, для забезпечення взаємодії та мобільності програмних додатків, даних і персоналу.

Мережна модель OSI (Open Systems Interconnection model), або еталона модель OSI – мережна модель стеку протоколів OSI/ISO (ДСТУ ІСО/МЕК 7498-1-99), що визначає правила взаємодії різних пристроїв у мережі.

Примітка *Модель OSI розроблена шляхом узагальнення досвіду, отриманого під час створення глобальних мереж (NPL, ARPANET, CYCLADES та ін.) у 70-ті рр. Її концепцію описано в роботі Чарльза Бахмана з компанії Honeywell Information Systems. Найповніший опис моделі становить понад 1000 сторінок тексту та визначає різні рівні взаємодії систем у мережах з комутацією пакетів, дає їм стандартні імена і розкриває функції, які повинен виконувати кожен рівень.*

Модель OSI визначає системні засоби взаємодії, що реалізує операційна система, системні утиліти, системні апаратні засоби. Модель не описує засоби взаємодії програм окремих користувачів (ці протоколи реалізуються шляхом звернення до системних засобів взаємодії). Тому в моделі OSI розрізняють:

1. Засоби взаємодії додатків (обчислювальне середовище) – це допоміжні процеси, призначені для обслуговування прикладних процесів, організації їх взаємодії. Програмне забезпечення, що реалізує ці функції, називають мережним.

2. Прикладні засоби – це процеси введення, збереження, оброблення та видачі інформації для потреби користувачів. Саме для виконання цих процесів створюють мережу.

Прикладні засоби різні, забезпечують різноманітність ресурсів, що надають інформаційні мережі. На відміну від них засоби взаємодії однакові, що дає змогу всім системам на однакових правах звертатися один до одного.

Модель OSI основана на поділі обчислювального середовища на сім рівнів, взаємозв'язок логічних об'єктів яких визначає відповідний набір семантичних і синтаксичних правил незалежно від будови рівня для кожної конкретної реалізації комп’ютерної технології (табл. 2.4).

Функції рівнів обчислювального середовища умовно можна поділити на дві групи:

– функції «media layers» – функції, які залежать від конкретної технічної реалізації мережі;

– функції «host layers» – функції, які не залежать від конкретної технічної реалізації мережі, а орієнтовані на роботу з додатками.

Таблиця 2.4

Структура моделі OSI (за даними ресурсу https://uk.wikipedia.org/wiki/)

Рівень	Тип даних	Функції	Основні протоколи	Обладнання
7. Прикладний (application)	Повідом-лення	Доступ до мережних служб	HTTP, FTP, POP3, SMTP тощо	Комп'ютер, під'єднаний до мережі (host), міжмережний екран (firewall)
6. Представ-ницький (presentation)		Шифрування та подання даних у потрібному форматі	ASCII, EBCDIC, JPEG, MIDI	
5. Сеансовий (session)		Керування сеансом зв'язку	RPC, PAP, L2TP тощо	
4. Транспорт-ний (transport)	Сегменти (segment) / датаграми (datagram)	Прямий зв'язок між кінцевими пунктами для надійного обміну даними	TCP, UDP, SCTP тощо	
3. Мережний (network)	Пакети (packet)	Визначення маршруту та логічна адресація	IPv4, IPv6, AppleTalk, ICMP тощо	Маршрутизатор, мережний шлюз
2. Канальний (data link)	Біти (bit), кадри (frame)	Фізична адресація	PPP, IEEE 802.22, Ethernet тощо	Мости, комутатори, точка доступу
1. Фізичний (physical)	Біти (bit)	Робота з середовищем передачі, сигналами та двійковими даними	USB, RJ	Концентратор, повторювач

Функції «media layers», або функції середовища передавання даних, притаманні для трьох нижніх рівнів – фізичного, канального, мережного, тобто мережно-залежних рівнів, протоколи яких тісно пов'язані з технічною реалізацією мережі та комунікаційним обладнанням, що тут застосовують.

Приклад *Функції «media layers» означають, що під час переходу на нове мережне обладнання, наприклад, FDDI повністю змінюють протоколи фізичного та канального рівнів всіх комп'ютерів, під'єднаних до цієї мережі.*

Комп'ютер з установленою на ньому мережною операційною системою взаємодіє з іншим комп'ютером за протоколами всіх семи рівнів через різні комунікаційні пристрої: концентратори, повторювачі, мости, комутатори, маршрутизатори тощо. При цьому залежно від типу комунікаційний пристрій може працювати тільки на фізичному рівні (повторювач), або на фізичному та канальному (міст), або на фізичному, канальному та мережному, іноді захоплюючи і транспортний рівень (маршрутизатор).

Транспортний рівень є проміжним, він приховує деталі функціонування нижніх рівнів від верхніх, що дає змогу розробляти програми, які не залежать від технічних засобів безпосереднього транспортування даних.

Функції «host layers» для трьох верхніх рівнів – прикладного, представницького, сеансового – мало залежать від технічних особливостей побудови мережі та спрямовані на вирішення завдань

надання прикладних сервісів на підставі наявної транспортної підсистеми. Вони орієнтовані на додатки, тому на них не впливають будь-які зміни в топології мережі, заміна мережного обладнання або перехід на іншу мережну технологію.

Приклад *Перехід від Ethernet до високошвидкісної технології 100VG-AnyLAN не потребує змін у програмних засобах, що реалізують функції бразузера або поштового клієнта.*

Таким чином, кожен рівень моделі OSI реалізує концептуально повну сукупність функцій в ієрархії функцій зв'язку відкритих систем, виконує певне завдання, пов'язане із забезпеченням обміну даними та обробленням даних в обчислювальній мережі. Це дає змогу будь-якій комп'ютерній технології реалізовувати властивості:

– розширюваності / масштабованості – забезпечення можливості додавання нових функцій (зміни наявних) при незмінних інших функціональних частинах технологій, а також побудови мережі з апаратних і програмних засобів різних виробників, що дотримуються єдиного стандарту;

– мобільності / переносності – забезпечення можливості перенесення програм і даних при модернізації / заміні апаратних платформ і роботи з ними фахівців без перепідготовки, безболісна заміна окремих компонентів мережі іншими, більш досконалими для її розвитку з мінімальними витратами;

– взаємодії – здатність до взаємозв'язку з іншими інформаційними комп'ютерними технологіями, об'єднаними мережею або мережами різного рівня – від локальної до глобальної;

– стандартизованості – проєктування комп’ютерних технологій на підставі узгоджених міжнародних стандартів, що реалізують відкритість у галузі інформаційних технологій для легкого з’єднання мереж;

– дружності до користувача – створення уніфікованого інтерфейсу, що дає змогу спокійно працювати користувачу без спеціальної комп'ютерної підготовки.

Модель OSI охоплює лише один аспект відкритості систем – відкритість засобів взаємодії пристроїв, під’єднаних до обчислювальної мережі. Тут під відкритою системою розуміють певний мережний пристрій, готовий взаємодіяти з іншим мережним пристроями за допомогою стандартних правил, які визначають формат, зміст і значення повідомлень, що приймають або надсилають.

Таким чином, реалізація й успішне впровадження розглянутих принципів до комп'ютерних технологій допомогли їх типізувати і сформувати кілька загальних ознак, що дають змогу:

– дотримуватися принципу відкритості систем (повинні бути сумісними з іншими платформами);

– реалізовувати механізми розподіленого оброблення даних;

– використовувати структуровані стандарти цифрового обміну даними та алгоритми, які визначають єдині правила передавання даних;

– широко використовувати комп'ютерне зберігання та надання інформації;

– передавати інформацію на практично безмежні відстані за допомогою цифрових технологій;

– забезпечувати підвищену стійкість до помилок і дотримання високих вимог до забезпечення цілісності даних.

Тестові запитання та завдання для самоперевірки

1. Архітектура комп’ютера – це:

а) технічний опис деталей пристроїв комп'ютера;

б) опис пристроїв для введення-виведення інформації;

в) опис алгоритму роботи програмного забезпечення комп'ютера;

г) опис принципів проєктування, роботи комп'ютера, достатній для розуміння користувача.

2. Арифметико-логічний пристрій – це елемент процесора, який:

а) виконує команди, що надходять на вхід, і керує роботою машини;

б) зберігає інформацію, яку часто використовують у роботі;

в) виводить текстову або графічну інформацію;

г) вводить алфавітно-цифрові дані.

3. Оперативна пам'ять необхідна:

а) для оброблення інформації;

б) для довготривалого зберігання інформації;

в) для зберігання програми та даних, з якими вона безпосередньо працює;

г) для запуску програмного забезпечення.

4. Установіть відповідність «визначення – термін» (наприклад, а – а, б – б та ін.):

Визначення	Термін
а) послідовність двійкових розрядів, розмірність якої збігається з розрядністю процесора	а) машинне слово
б) послідовність двійкових розрядів коду інструкції, яка виконується машиною автоматично без будь-яких додаткових указівок і пояснень	б) стандартне машинне слово
в) послідовність двійкових розрядів фіксованої довжини для збереження у комірці, що сприймається процесором як єдине ціле	в) операнд
г) n-розрядні двійкові послідовності, що беруть участь у арифметичних і логічних перетвореннях	г) машинна команда

5. «Програму слід подавати в пам'яті комп'ютера в цифровій формі разом з даними», – писав Джон фон Нейман у науковій роботі. Це відповідає принципу:

а) двійкового кодування;

б) адресності;

в) однорідності пам’яті;

г) програмного керування;

д) ієрархії.

6. Розрядність шини даних пов'язана:

а) з розрядністю машинного слова, що обробляється процесором;

б) з величиною адресного простору процесора;

в) з розрядністю поля адреси машинної команди;

г) з розрядністю пристрою керування арифметико-логічного пристрою.

7. Установіть відповідність «суть – принцип» (наприклад, а – а, б – б та ін.):

Суть	Принцип
а) команди та дані зберігають в пам'яті комп'ютера та ззовні в ній не розрізняються	а) двійкового кодування
б) вирішення всіх завдань в комп'ютерах здійснюється програмним способом	б) адресності
в) всю інформацію в комп'ютері кодують двійковими бітами	в) однорідності пам'яті
г) пам'ять комп'ютера складається з пронумерованих комірок, в які записують коди команд і даних	г) програмного керування

8. Виберіть правильне визначення:

а) регістр – логічний пристрій для фізичної реалізації комірки пам'яті всередині арифметико-логічного пристрою;

б) регістр – лічильник команд у пристрої керування арифметико-логічного пристрою;

в) регістр – елементарна інструкція, яка виконується машиною автоматично без будь-яких додаткових указівок і пояснень.

9. Що таке «hardware»:

а) це найпопулярніша операційна система для комп'ютерів IBM PC;

б) це апаратна частина комп'ютера;

в) це операційна система, яка забезпечує створення нових програм;

г) це модернізація апаратної або програмної частини комп'ютерів?

10. Виберіть правильне визначення:

а) операційна система – це набір правил, що визначають вимоги до зберігання даних, які найчастіше використовуються процесором;

б) операційна система – це елемент програмного керування комп'ютера, призначений для того, щоб показати користувачеві роботу апаратного забезпечення комп'ютера;

в) операційна система – це єдині правила побудови машинних команд, орієнтованих на керування комп'ютером;

г) операційна система – це комплекс системних програм, призначених для керування або оброблення, які реалізують найважливіші функції.

11. Згідно з моделлю OSI установіть відповідність «функція – рівень» (наприклад, а – а, б – б та ін.):

Функція	Рівень
а) робота з середовищем передавання даних	а) прикладний
б) визначення маршруту	б) представницький
в) шифрування та подання даних у потрібному форматі	в) мережний
г) доступ до мережних служб	г) фізичний

12. Сучасні комп'ютерні технології мають набір типових властивостей. Установіть відповідність «суть – властивість» (наприклад, а – а, б – б та ін.):

Суть	Властивість
а) забезпечення можливості перенесення програм і даних при модернізації або заміні апаратних платформ	а) взаємодії
б) проєктування і розроблення інформаційних технологій на основі узгоджених міжнародних стандартів	б) мобільності
в) забезпечення можливості додавання нових функцій або зміни деяких наявних при незмінних інших функціональних частинах мережних технологій	в) розширюваності
г) здатність до взаємозв'язку з іншими інформаційними комп'ютерними технологіями, що поєднані мережами різних рівнів	г) стандартизованості

13. Принцип відкритої архітектури означає:

а) що комп'ютер зроблено єдиним нерозбірним пристроєм;

б) можлива легка заміна застарілих частин персонального комп'ютера;

в) що комп’ютер має спеціалізований інтерфейс, який дає змогу працювати спеціально підготовленому користувачу;

г) що заміна однієї деталі комп’ютера призводить до заміни всіх його елементів.

14. Скільки можна подати цілих чисел без знака за умови використання процесора Intel 80486? Укажіть можливий діапазон цих цифр.

15. Архітектура процесора XScale передбачає використання 32-розрядного машинного слова. Знайдіть інформаційну місткість для визначення кількості цілих чисел без знака, які можна подати за допомогою цього слова. Укажіть можливий діапазон цих цифр.

16. Український алфавіт містить 33 літери. Яка мінімальна кількість розрядів потрібна для подання в комп'ютері букв українського алфавіту. Чи зміниться розрядність при поданні літер англійської мови, якщо її алфавіт складається з 26 літер?

17. Розшифруйте машинні команди, подані мовою символічного кодування, які надійшли на вхід арифметично-логічного пристрою моделі 74181:

1001 0	0010	0101	1011

0101 1	0110

1011 1	0111	1001

.

18. Згідно з протоколом Modbus RTU сформуйте запит від master-пристрою до slave-пристрою з адресою 0x05. Слід зчитати дані з регістрів з адресами 0x200B і 0x200C, які надійшли на вхід slave-пристрою. Значення контрольної суми вказати довільно.

19. Відповідно до протоколу Modbus RTU сформуйте запит master-пристрою на запис значень до 20 регістрів зберігання даних, починаючи з регістра з адресою 0x011E, які надійшли на вхід slave-пристрою з адресою 0x0C. Значення контрольної суми вказати довільно.

Лекція 3. ПОДАННЯ ДАНИХ У КОМП'ЮТЕРНИХ ТЕХНОЛОГІЯХ

План

1. Машинні системи числення.
2. Подання текстової та символьної інформації в комп'ютерах.
3. Подання числової інформації в комп'ютерах.

1. Машинні системи числення

Для успішної взаємодії функціональної та системної частин комп'ютерів, забезпечення сумісності програмних та апаратних засобів усі дані слід подавати в однаковому вигляді. Тому вивчення комп'ютерних технологій почнемо із вивчення методів подання даних, зокрема машинного подання, що є корисними у випадках [1, 24]:

1. Налагодження програм, які працюють з двійковими файлами, під час перегляду двійкових файлів на фізичному рівні.

2. Керування пам'яттю на фізичному рівні, коли є необхідним доступ до «складових частин» формату даних, що є актуальним під час переходу від одного типу даних до іншого при одночасному збереженні вмісту, тощо.

Дані, які мають зміст і придатні для оброблення або використання користувачами комп'ютерних технологій, називають повідомленнями. Саме повідомлення, тобто деяка форма подання вихідної інформації, розглядають як операнди сеансового, представницького та прикладного рівнів (див. табл. 2.4).

Будь-який ГІС-додаток використовує повідомлення, які перетворює на вихідні дані за певними алгоритмами. Ці дані подаються у формі, зручній для роботи людини, наприклад, числа зображуються в десятковій системі числення, текст – у вигляді слів, складених з букв алфавіту. Під час введення даних здійснюється перетворення повідомлень на числову послідовність, яка згідно з принципом фон Неймана подається у двійковому коді. Тому двійкова система числення стала «рідною» для комп'ютерів, які працюють тільки з сигналами. При цьому вважають, що один із двох станів сигналу означає одиницю, інший – нуль. Таким чином, оброблення даних у ГІС-додатках – це перетворення чисел на нулі й одиниці за деякими правилами. Тому робота будь-якої комп'ютерної технології пов'язана з поняттям числа і двійкової системи числення [24].

Під системою числення розуміють сукупність прийомів іменування та позначення чисел. Будь-яке число можна записати послідовністю умовних знаків – цифрами. Крім значення цифри також важливою є її позиція, що визначає вагу (саме тому системи числення називають позиційними) [1].

Величину, на яку наступна цифра «важча» за попередню, називають основою системи числення ρ. У системі числення з основою ρ використовують цифри від 0 до ρ-1. Вони становлять базу системи числення.

Нехай є множина символів $\{a_n, a_{n-1}, ..., a_1, a_0, a_{-1}, ..., a_{-m}\}$, де кожен з

символів означає ρ-ту цифру. Тоді запис числа має вигляд

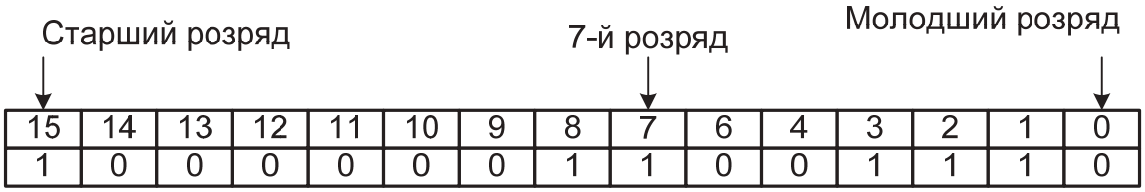
$$a_n a_{n-1}...a_1 a_0, a_{-1}...a_{-m}, \qquad (3.1)$$

де кома поділяє число на дві частини – цілу та дробову; індекси вказують на позицію цифр, що утворюють число.

Позиції, розташовані лівіше від коми, перенумеровані справа наліво числами 0, 1, 2 тощо, утворюють цілу частину числа. Позиції, розташовані праворуч від коми, пронумеровані зліва направо з використанням чисел -1, -2 та ін., становлять дробову частину числа.

Перенумеровані позиції числа називають розрядами. Молодший розряд – це початок координат, з якого починається виконання всіх операцій.

Тому в пам'яті комп'ютера числові дані зберігають, починаючи з молодших розрядів (рис. 3.1).



15	14	13	12	11	10	9	8	7	6	4	3	2	1	0
1	0	0	0	0	0	0	1	1	0	0	1	1	1	0

Рис. 3.1. Нумерація розрядів під час зберігання цілих чисел

Цифра, яка знаходиться в нульовому розряді, має значення, що відповідає числу бази. Цифра, яка розташована в деякому розряді, має значення в ρ разів більше за те, що вона мала б у розряді з номером, меншим на одиницю (або значення в ρ разів менше за те, яке вона мала б у розряді з номером, більшим на одиницю). Звідси випливає таке правило:

послідовність ρ-х цифр позначає число, що дорівнює сумі значень його цифр.

Тому в позиційній системі числення число (3.1) можна записати відповідно до формули [1, 4]

$$a_n a_{n-1}...a_1 a_0, a_{-1}...a_{-m} = a_n \rho^n + a_{n-1}\rho^{n-1}... +$$
$$+...a_1\rho^1 + a_0\rho^0 + a_{-1}\rho^{-1} + ... + a_{-m}\rho^{-m} = \sum_{i=n}^{-m} a_i \rho^i, \qquad (3.2)$$

де a_i – коефіцієнти, які при окремих розрядах числа набувають значень 0, 1, …, ρ-1; ρ – основа системи числення; –m – кількість цифр дробової частини числа; n – кількість цифр цілої частини числа.

Приклад. З використанням виразу (3.2) подати число $8401{,}302_{10}$ у вигляді суми.

Вихідне число подано в десятковій системі числення, тому $\rho = 10$. Відповідно до виразу (3.2) отримаємо

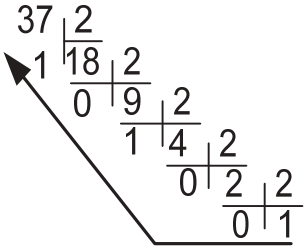
$$8401{,}302_{10} = 8\cdot 10^3 + 4\cdot 10^2 + 0\cdot 10^1 + 1\cdot 10^0 + 3\cdot 10^{-1} + 0\cdot 10^{-2} + 2\cdot 10^{-3}.$$

Для подання чисел у різних системах числення припускається їх однозначне переведення з однієї системи числення в іншу. Переведення

числа з ρ-ї системи числення в q-ту, коли $\rho \neq q^k$, здійснюють за умови окремого переведення цілої та дробової частин числа.

Правило переведення цілої частини числа. Ціле число q записують у ρ-й системі числення. Ділять q на ρ у q-й системі числення. В остачі отримують число, яке визначає останню цифру необхідного ρ-го запису. Одержану частку знов ділять на ρ, в остачі отримують число, що визначає передостанню цифру необхідного ρ-го запису. Процес повторюють доти, доки в частці не отримають число, менше за ρ. Воно визначить першу цифру ρ-го запису.

Приклад. Записати в двійковій системі числення число 37_{10}.

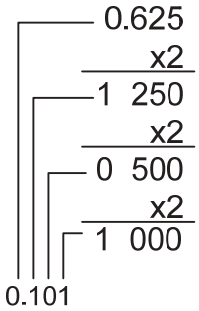


Відповідь: $37_{10} = 100101_2$

Правило переведення дробової частини числа. Число q записують у ρ-ту систему числення. Множать у q-й системі числення дріб на ρ. Одержана ціла частина добутку визначає першу цифру необхідного ρ-го запису дробу. Дробову частину добутку знов множать на ρ. Одержана ціла частина добутку визначає наступну цифру необхідного ρ-го запису дробу. Процес повторюють доти, доки не одержать цілий добуток або не отримають необхідну кількість цифр ρ-го запису дробу.

Приклад. Перевести в двійкову систему числення число $0{,}625_{10}$.

Виконаємо такі дії:



Через незручність і незрозумілість для людини процесу ділення в інших системах числення переведення чисел з цих систем в загальноприйняту десяткову здійснюють з використанням формули (3.2). При цьому в її правій частині ρ-ті цифри, безпосередньо число ρ і показник степеня, записують в десятковій системі числення, далі знаходять значення одержаного виразу за правилами, прийнятими в десятковій системі числення.

Приклад. Перевести в десяткову систему числення число 10111_2.

Отримаємо

$$10111_2 = 1\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 23_{10}.$$

У сучасних комп'ютерних додатках широко використовують конвертори для переведення цілих десяткових чисел в інші системи числення і навпаки. Наприклад, в комп'ютерній системі Scilab використовують оператори такого вигляду [25]:

– **dec2bin**(X), **bin2dec**(X) – пряма, зворотна функції переведення цілого беззнакового числа X із десяткової системи числення в двійкову і із двійкової системи числення в десяткову;

– **dec2hex**(X), **hex2dec**(X) – пряма, зворотна функції переведення цілого беззнакового числа X із десяткової системи числення в шістнадцяткову і навпаки;

– **dec2oct**(X), **oct2dec**(X) – пряма, зворотна функції переведення цілого беззнакового числа X із десяткової системи числення в вісімкову і навпаки.

Зазначимо, що деякі завдання, наприклад економічні розрахунки, характеризуються великими обсягами вхідної та вихідної інформації при порівняно невеликому обсязі обчислень. Переведення таких чисел із десяткової системи числення в двійкову для подальшого оброблення і назад (для виведення результату обчислень) потребує 20 – 40 машинних команд. Тому для прискорення процесу оброблення інформації передбачено можливість виконання операцій над числами, поданими з використанням прямого або двійково-десяткового кодування (особливо це поширено у мейнфреймах, де інформація, яка вводилася з перфокарт, подавалася в двійково-десятковому коді).

У загальному випадку двійково-кодовані системи числення утворюються в такий спосіб [22].

Нехай ρ – основа позиційної системи числення. Однозначної відповідності між ρ-ми цифрам і не рівними між собою цілими двійковими цифрам можна досягти за таким правилом:

визначити кількість розрядів k найбільшої ρ-ї цифри, подати її у двійковій системі числення та вирівняти за нею розрядності інших цифр шляхом допису до їх двійкового подання необхідної кількості нулів зліва.

Отже, формується k-розрядне двійкове подання цифри, яке називають її двійковим кодом. При цьому найменшу необхідну розрядність цих кодів знаходять за умови

$$2^{k-1} < \rho \le 2^k. \qquad (3.3)$$

З виразу (3.3) кількість k-розрядних двійкових цифр, які не використовують як коди для ρ-х цифр, дорівнює

$$2^k - \rho. \qquad (3.4)$$

Ці цифри визначають множину заборонених комбінацій нулів та одиниць.

Сформуємо двійково-десяткову систему числення.

Для десяткових цифр за виразом (3.3) знайдемо необхідну кількість розрядів їх двійкових кодів:

$$k \geq \log_2 \rho \geq \log_2 10 = 4,$$

тобто для кодування десяткових цифр (бази десяткової системи числення) необхідна сукупність тетрад (група з чотирьох двійкових розрядів), що визначає їх двійковий код.

Відповідність між цифрами десяткової системи числення та їх двійковим кодом наведено в табл. 3.1. Ці цифри утворюють базу двійково-десяткової системи числення та становлять множини дозволених комбінацій.

Таблиця 3.1

Двійково-десяткове подання чисел

Вихідний символ	Дозволені комбінації	Вихідний символ	Дозволені комбінації
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

За виразом (3.4) кількість заборонених комбінацій становить:

$$2^k - \rho = 2^4 - 10 = 6.$$

Це комбінації, що знаходяться в діапазоні (1010_2 … 1111_2) і кодують числа (10_{10} … 15_{10}). З цих комбінацій, наприклад, вибирають коди знака для машинного подання чисел.

Для зображення десяткового числа потребується стільки тетрад, скільки є десяткових розрядів у числі. Звідси випливає таке правило:

десяткові цифри подають у двійковій системі числення, а всі розряди числа без зміни – у десятковій системі числення.

Приклад. Подати десяткове число $A_{10} = 893{,}2$ у двійково-десятковому вигляді.

З використанням табл. 3.1 отримаємо двійкові тетради для кожної цифри вихідного числа: 8 – 1000; 9 – 1001; 3 – 0011; 2 – 0010. Сформуємо двійково-десятковий запис:

$$A_{2\text{-}10}=1000\ 1001\ 0011,\ 0010.$$

Серед переваг двійково-десяткового кодування, крім зручного та зрозумілого процесу переведення з десяткової системи числення, слід зазначити те, що цей код дає змогу виконувати арифметичні операції в десятковій системі числення, використовуючи двійкові елементи для зберігання та перероблення числової інформації [21].

Елемент двійкової послідовності коду даних називають бітом [25].

Послідовність із восьмі бітів називають байтом.

Байт – основна одиниця виміру інформації в комп'ютерах, машинне

слово мінімальної розмірності, яке використовують під час роботи з даними. Байт – величина, прийнята як стандарт зберігання та передавання даних каналами зв'язку, подання текстової інформації. Крім того, байт – це універсальний «вимірювальний інструмент» – розмірність усіх форм подання даних установлюють кратною байту [25, 26]. При цьому стандартне машинне слово вважають розбитим на байти, які нумерують, починаючи з молодших розрядів (див., наприклад, рис. 3.1).

Під час роботи з двійковою інформацією застосовують декілька термінів для позначення сукупності двійкових розрядів, які використовують для виміру місткості інформації, що зберігається або обробляється комп'ютерами (табл. 3.2) [26].

При цьому кратні префікси для створення похідних від байта одиниць виміру інформації формують таким чином:

– зменшувальні префікси не використовують;

– одиниці виміру інформації, менші за байт, називають спеціальними словами (ніббл і біт);

– у наявних стандартах немає скорочення для «біт», тому запис «Гб»* не є синонімом «Гбіт», але у міжнародному стандарті МЕК рекомендують позначення: «bit» – для біта; «o» або «B» – для байта (причому «o» – єдине зазначене позначення французькою мовою).

Таблиця 3.2

Одиниці виміру інформації, кратні байту

ДСТУ 8.417-2002 (погоджено з системою SI)			МЕК IEC 60027-2 2005		
Назва	Символ	Степінь	Назва	Символ	Степінь
Байт	Б	10^0	Байт	B / Б	2^0
Кілобайт	КБ	10^3	Кібібайт	KiB / КіБ	2^{10}
Мегабайт	МБ	10^6	Мебібайт	MiB / МіБ	2^{20}
Гігабайт	ГБ	10^9	Гібібайт	GiB / ГіБ	2^{30}
Терабайт	ТБ	10^{12}	Тебібайт	TiB / ТіБ	2^{40}
Петабайт	ПБ	10^{15}	Пебібайт	PiB / ПіБ	2^{50}
Ексабайт	ЕБ	10^{18}	Ексбібайт	EiB / ЕіБ	2^{60}
Зетабайт	ЗБ	10^{21}	Зебібайт	ZiB / ЗіБ	2^{70}
Йотабайт	ЙБ	10^{24}	Йобібайт	YiB / ЙіБ	2^{80}

Міжнародна електротехнічна комісія (МЕК) рекомендує використовувати двійкові префікси, але на практиці їх поки не застосовують, можливо, через неблагозвучність – кібібайт, мебібайт, йобібайт тощо.

Негласно існує таке правило:

$$2^{10} = 1024 \approx 1000 = 10^3,$$

але нескладно помітити, що різниця між величинами, вираженими в кіло

* Відповідно до вимог ДСТУ для позначення байта використовують велику літеру «Б» з метою уникнення плутанини між скороченнями від байт і біт.

(тобто 10^3 = 1000) і в кібі (тобто 2^{10} = 1024), зростає зі збільшенням ваги префікса. При порівнянні розміру 100 КБ зі 100 КіБ (табл. 3.3) різниця відносно невелика (близько 2,34 %). Однак зі збільшенням розміру даних ця різниця зростає: при порівнянні розміру 100 ЗБ зі 100 ЗіБ різниця становить більше 15,3 %. Адже чим більшим є обсяг даних, тим більшою стає різниця між фактичним та позначеним розміром інформації, між фактичною і позначеною місткостями запам'ятовувальних пристроїв тощо.

Примітка *Багато виробників для визначення розміру даних, а отже, інформаційної місткості, використовують десяткові та двійкові одиниці. Наприклад, компанія IBM Spectrum Control для визначення розмірів пам'яті та дискового простору використовує основу 2, рекомендовану МЕК, і основу 10, погоджену з системою SI, – для визначення місткості фізичних жорстких дисків.*

Таблиця 3.3

Різниця між десятковими та двійковими значеннями одиниць виміру розміру даних

Десяткове значення	Двійковий еквівалент	Різниця
100 Кілобайт (КБ)	97,66 Кібібайт (КіБ)	2,34 %
100 Мегабайт (МБ)	95,37 Мебібайт (МіБ)	4,63 %
100 Гігабайт (ГБ)	93,13 Гібібайт (ГіБ)	6,87 %
100 Терабайт (ТБ)	90,95 Тебібайт (ТіБ)	9,05 %
100 Петабайт (ПБ)	88,82 Пебібайт (ПіБ)	11,18 %
100 Ексабайт (ЕБ)	86,74 Ексбібайт (ЕіБ)	13,26 %
100 Зетабайт (ЗБ)	84,70 Зебібайт (ЗіБ)	15,30 %
100 Йотабайт (ЙБ)	82,72 Йобібайт (ЙіБ)	17,28 %

Приклад. Знайти фактичну місткість жорсткого диска, якщо компанія-виробник заявляє, що накопичувач має місткість 4 ТБ.

За даними табл. 3.3 відомо співвідношення 100 ТБ ≈ 90,95 ТіБ, отже, фактична місткість жорсткого диска становить 3,64 ТіБ, тобто на ньому можна зберігати 3,64 ТіБ двійкових даних.

Також пригадаємо, що залежно від рівня моделі OSI протокольний блок даних називають по-різному (див. табл. 2.4): на фізичному рівні розглядають біти, на канальному – кадри даних, на мережному – пакети або датаграми, на транспортному – сегменти тощо.

Примітка *Поряд з фізичними світовими константами (сталою Планка, швидкістю світла, гравітаційною сталою) виділяють декілька інформаційних констант, які накладають фундаментальні обмеження на характеристики інформаційних систем. Тому обсяг інформації в природних системах обмежено величиною інформаційної межі, значення якої становить 10^{90} бітів.*

Оброблення даних усередині комп'ютера – це перетворення слів з нулів та одиниць за правилами, зафіксованим у мікросхемах процесора. Але зазначимо, що за простоту та зручність роботи з двійковою системою доводиться платити: чим менша основа системи числення, тим більше цифр необхідно для подання вихідного числа. Отже, у двійковій системі

число має максимальну розрядність [25]. Користуватися такими числами через їх велику довжину та візуальну однорідність людині вкрай незручно. Наприклад, IP-адреса

128.10.2.15

у двійкової системі числення має вигляд

10000000 00001010 00000010 00001111.

Тому програмісти та інженери на етапах створення програм для комп'ютерів, їх налагодження, ручного введення / виведення даних, а також на етапах розроблення, створення та налаштування обчислювальних систем замінюють коди машинних команд, адреси й операнди на еквівалентні їм величини в так званих «машинних» системах числення [11].

До «машинної» групи систем числення відносять [1]:

– двійкову систему числення;

– вісімкову систему числення;

– шістнадцяткову систему числення.

Приклад *При створенні та налаштуванні LAN використовують апаратні адреси – адреси мережного адаптера або порту маршрутизатора для ідентифікації пристрою в межах LAN, тобто MAC-адреси. Ці адреси призначають виробники обладнання, вони унікальні, видаються централізовано, складаються з 6 байтів, які записані у шістнадцятковій системі числення, наприклад, 11-A0-17-3D-BC-01.*
Вісімкову систему застосовують під час визначення прав доступу до файлів і прав виконання для учасників у Linux-системах, а також при описанні прав на файли в стилі Unix/POSIX (0666, 0750) або в окремих випадках роботи з бітовими масками. Але застосування вісімкових констант у програмах, наприклад мовою Сі, є надзвичайно низьким (близько 0,1 % від усіх літеральних констант – констант, безпосередньо розміщених у коді).

Зазначимо, що вісімкову систему числення раніше використовували в програмуванні та комп'ютерній документації, але сьогодні її майже повністю витіснила шістнадцяткова.

Приклад *Офіційне народження двійкової арифметики пов'язане з ім'ям Г. В. Лейбніца, який у 1703 р. опублікував статтю, де сформував правила виконання арифметичних операцій над двійковими числами.*
В історії відомо курйозний випадок із вісімковою системою числення. Шведський король Карл XII у 1717 р. захопився цією системою числення, вважав її зручнішою за десяткову, і мав намір королівським указом впровадити її як загальноприйняту. Несподівана смерть завадила королю здійснити такий незвичайний намір.

Перевагами «машинних» систем числення є [1]:

1. Можливість компактного подання двійкового запису числа, тому що цей запис у вісімковій або шістнадцятковій системі числення буде відповідно в три або чотири рази коротшим за двійковий.

2. Порівняно спрощене переведення чисел із двійкової системи числення в вісімкову або шістнадцяткову і навпаки.

Алгоритм переведення чисел з двійкової системи числення в вісімкову або шістнадцяткову такий:

1. Подати число сукупністю тріад (групою з трьох двійкових розрядів) у випадку переведення числа в число вісімкової системи числення або сукупністю тетрад – у число шістнадцяткової системи числення. Для цього, починаючи з молодшого розряду, поділити двійкову послідовність числа на тріади (тетради), доповнюючи за необхідності нулями крайні групи.

2. Кожну групу (тріаду або тетраду) замінити відповідною вісімковою або шістнадцятковою цифрою (табл. 3.4).

Таблиця 3.4

Переведення чисел у різні системи числення

Система числення				Система числення			
Десят-кова	Двійко-ва	Вісімко-ва	Шістнад-цяткова	Десят-кова	Двійко-ва	Вісімко-ва	Шістнад-цяткова
0	0000	0	0	8	1000	10	8
1	0001	1	1	9	1001	11	9
2	0010	2	2	10	1010	12	A
3	0011	3	3	11	1011	13	B
4	0100	4	4	12	1100	14	C
5	0101	5	5	13	1101	15	D
6	0110	6	6	14	1110	16	E
7	0111	7	7	15	1111	17	F

Приклад. Перевести двійкове число 10101111_2 у вісімкову систему числення.

Поділимо вихідне число на тріади, починаючи з молодшого розряду:

10 101 111.

Ураховуючи, що старші розряди не утворюють тріади, доповнимо їх нулем справа:

010 101 111.

За даними табл. 3.4 визначимо вісімкове значення кожної тріади окремо, тобто двійкове число $010_2 = 2_8$, число $101_2 = 5_8$, а число $111_2 = 7_8$.

Отже, двійковому числу 10101111_2 еквівалентним є вісімкове число 257_8.

Приклад. Перевести в шістнадцяткову систему числення IP-адресу 128.10.2.15.

Поділимо двійкову IP-адресу на тетради, з використанням табл. 3.4 кожній тетраді поставимо у відповідність шістнадцяткову цифру. Отримаємо

1000 0000 0000 1010 0000 0010 0000 1111
8 0 0 A 0 2 0 F

тобто IP-адреса 128.10.2.15 у шістнадцятковій системи числення матиме вигляд 80.0A.02.0F.

Приклад *Нові стандарти глобальної адресації, зокрема протокол IPv6, для позначення адрес використовують шістнадцяткові цифри, записані у вигляді восьми груп з чотирьох цифр, де кожну групу розділяє двокрапка. Тому типова IPv6-адреса має такий вигляд: 2001:0db8:85a3:08d3:1319:8a2e:0370:7334.*

Таким чином, вісімкова і шістнадцяткова системи числення стають найпростішою мовою спілкування людини з комп'ютером: вони досить близькі до зручної для людини десяткової системи числення та до двійкової «мови» машини.

При введенні числових даних для розрізнення того, в якій системі числення подано константи, застосовують спеціальні позначення:

– шістнадцяткова константа складається з набору шістнадцяткових цифр з префіксом «0x»;

– вісімкова константа складається з цифр з набору від 0 до 7 і починається з 0.

Приклад *Деякі мови програмування, зокрема C/C++, Ada, Perl, Java, Python (до версії 3.0), використовують синтаксис (аналогічний використанню префікса «0x» для шістнадцяткових констант) для запису вісімкових констант із застосуванням ведучого нуля.*

Приклад. Записати двійкову константу $B_2=111110101_2$ у вісімковій і шістнадцятковій системах числення.

Поділимо константу на тріади та тетради і з використанням табл. 3.4 поставимо їм у відповідність:

– вісімкові цифри

111 110 101
7 5 6,

тобто $B_2 = 111110101_2$ відповідає вісімкова константна 0765;

– шістнадцяткові цифри

0001 1111 0101
1 F 5,

тобто $B_2 = 111110101_2$ відповідає шістнадцяткова константна 0x1F5.

Розглянемо отримане шістнадцяткове число B = 0x1F5. Оскільки позначення складається з восьми цифр, то в пам'яті машини це число слід розміщувати побайтово таким чином: 01_F5. Насправді в будь-якій дампі (dump – виведення вмісту пам'яті у внутрішньому поданні) це число буде записано молодшими байтами вперед, тобто F5_01. При цьому перевертається не весь dump, а тільки та частина, яка є записом числа. Таким чином виявляється фундаментальна властивість пам'яті:

за її формальним вмістом не можна визначити, що там розміщено. Все залежить від формату – послідовності розміщення даних та їх інтерпретації.

Наприклад, байт 0xC2 інтерпретують у форматі цілого числа зі знаком як від'ємне число -62, без знака – як додатне число 194, у системі кодування Windows-1251 – як символ текстового файлу "В" або просто як

машинне слово, що складається з нулів та одиниць.

Отримавши дані, комп'ютери за певними алгоритмами «механічно» перетворюють їх на вихідні дані (результати), які подать у зручному для людини вигляді, наприклад, числа зображують у десятковій системі числення, текстову інформацію – у вигляді слів, що складаються з букв, тощо. Відображення символів вхідних даних у внутрішньому поданні називається кодуванням, де під кодом розуміють множину слів (кодових комбінацій), яку при цьому використовують.

Важливо запам'ятати таке правило:

> коди окремих значень, що кодують різні види інформації (числа, текст, графіку, звук), можуть збігатися, тому декодування кодованих даних здійснюють за контекстом при виконанні окремих команд програми.

Конкретну відповідність між символами інформації та їх кодами називають системою кодування. Системи кодування значно залежать від стандартів, що застосовують у кожній країні, типу діючого обладнання та ін. [8].

2. Подання текстової та символьної інформації в комп'ютерах

Правила запису бітів у послідовність визначає відповідний формат залежно від типу даних. Послідовність бітів у форматі, що має певний зміст, називають полем.

Текстову інформацію, як і будь-яку іншу, зберігають у пам'яті комп'ютера в двійковому вигляді – кожному символу ставлять у відповідність деяке додатне двійкове число, яке називають кодом символу.

Під час натискання будь-якої клавіші при введенні текстової інформації контролер клавіатури виробляє два типи кодів – scan-коди та віртуальний код клавіші.

Scan-код – код, наданий кожній клавіші, за допомогою якого драйвер клавіатури розпізнає, яку з клавіш натискали [27].

В IBM-сумісних комп'ютерах при натисканні будь-якої клавіші контролер клавіатури розпізнає клавішу та залежно від формату надсилає її scan-код – певне шістнадцяткове число (табл. 3.5) – на порт 60_{16}. При відпусканні клавіші контролер клавіатури більш старого формату IBM PC/XT посилає на той самий порт scan-код, збільшений на 80_{16}, а нового формату IBM PC/AT – два байти: $F0_{16}$ і scan-код.

Більшість scan-кодів формату XT відповідають фізичному розташуванню клавіш, починаючи з Esc (код 01_{16}) і цифрових клавіш 1 – 9 (коди 02_{16} – $0A_{16}$). Наприклад, клавіші другого ряду клавіатури мають послідовні scan-коди $0F_{16}$ – $1C_{16}$ (15 – 28 у десятковій системі) тощо. Зазначимо, що в усіх операційних системах фірми Microsoft scan-коди формату AT перетворюються на scan-коди формату XT, а отже, всі програми отримують вже scan-коди набору XT.

Scan-коди жорстко прив'язані до кожної клавіші на апаратному рівні та не залежать від стану індикаторів CapsLock, ScrollLock, NumLock, а

також стану керувальних клавіш Shift, Alt, Ctrl.

Таблиця 3.5

Scan-коди деяких клавіш у шістнадцятковій системі числення

Клавіша	Коди формату *IBM PC/XT*		*Коди формату IBM PC/AT*	
	Натискання	Відпускання	Натискання	Відпускання
A	1E	9E	1C	F0,1C
L	26	A6	4B	F0,4B
1	02	82	16	F0,16
[	1A	9A	54	F0,54
End	E0,4F	E0,CF	E0,69	E0,F0,69
PageUp	E0,49	E0,C9	E0,7D	E0,F0,7D
Delete	E0,53	E0,D3	E0,71	E0,F0,71
PrintScreen	E0,2A,E0,37	E0,B7,E0,AA	E0,12,E0,7C	E0,F0,7C,E0,F0,12
Pause	E1,1D,45,E1,9D,C5		E1,14,77,E1,F0,14,FO,77	

Після визначення того, яку клавішу натиснуто, контролер клавіатури виробляє службові коди, які передає для подальшого оброблення на рівні BIOS і Windows, Linux тощо.

Службовий або віртуальний код клавіші – двійкове число, що ідентифікує символ, причому віртуальний код клавіші не пов'язаний з формою символу, який на неї нанесено.

Визначення символу та присвоєння йому внутрішнього віртуального коду здійснює спеціальна програма комп’ютера за спеціальними таблицями (ASCII, EBCDIC, GBK, Unicode та ін.), які залежно від типу операційної системи та конкретних прикладних програм використовують 8- або 16-розрядні коди символів.

Для подання латиниці традиційно використовували дві основні системи кодування [28]:

– ASCII (American Standard Code for Information Interchange – американський стандартний код обміну інформацією), що поширений у сучасних персональних комп'ютерах;

– EBCDIC (Extended Binary Coded Decimal Information Code – розширений двійково-десятковий інформаційний код), який застосовують у комп’ютерних системах (мейнфеймах) фірми IBM, наприклад AS/400, System/360, System/370, System/390, z 90, як розвиток кодів перфокарт.

Спочатку стандарт ASCII, введений у 1963 р., ставив у відповідність кожному символу семирозрядний двійковий код, застосовуючи для подання 7 бітів / символ, що було ґрунтовним для кодування мов алфавітної писемності (латиниця, кирилиця, арабський алфавіт, грецька мова тощо).

Пізніше відповідно до стандарту ISO 8895-1 систему ASCII розширили з можливістю кодування 256 різних символів, застосовуючи для їх подання 8-розрядний двійковий код. При цьому згідно з властивістю розширюваності структурно стандарт поділено на дві частини (його подають у вигляді таблиці кодів):

1. Базова частина таблиці стандарту (табл. 3.6), яка визначає відповідність між символами та їх кодами в діапазоні від 0 до 127 із

застосуванням послідовного кодування, коли букви розташовано за алфавітом, а цифри – за зростанням, що дає змогу визначати код символу, не використовуючи таблицю. При цьому коди цифр беруть з цієї таблиці тільки при введенні (або виведенні) за умови, що їх використовують як елемент тексту. У випадку, коли цифри беруть участь у обчисленнях, їх переводять у двійкову систему числення за розглянутими раніше правилами.

Таблиця 3.6

Базова частина таблиці ASCII

Код стовпця	Код рядка									
	0	1	2	3	4	5	6	7	8	9
…	…	…	…	…	…	…	…	…	…	…
3	…	…	пробіл	!	“	#	$	%	&	‘
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	‘	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	–			

Перші 32 коди базової частини таблиці, починаючи з нульового, надано виробникам апаратних засобів для формування керувальних кодів, тобто вони не відповідають жодному символу мови. Починаючи з 32 по 127 коди, містяться коди для символів англійського алфавіту, розділових знаків, арифметичних дій, цифр і деяких допоміжних символів.

2. Розширена частина таблиці стандарту, яка визначає відповідність між символами національних шрифтів, псевдографіки, спеціальних математичних знаків та їхніми кодами в діапазоні від 128 до 255 (табл. 3.7). Цю частину таблиці можна замінювати за допомогою відповідних драйверів для застосування в одному документі декількох шрифтів і гарнітур.

Таблиця 3.7

Фрагмент таблиці стандарту ASCII

Код стовпця	Код рядка									
	0	1	2	3	4	5	6	7	8	9
16	…	…	…	Ј	¤	Ґ	¦	§	Ё	©
17	Є	«	¬	-	®	Ї	°	±	І	і
18	ґ	µ	¶	·	ё	№	є	»	ј	S
19	s	ї	А	Б	В	Г	Д	Е	Ж	З
20	И	Й	К	Л	М	Н	О	П	Р	С
21	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы
22	Ь	Э	Ю	Я	а	б	в	г	д	е
23	ж	з	и	й	к	л	м	н	о	п
24	р	с	т	у	ф	х	ц	ч	ш	щ
25	ъ	ы	ь	э	ю	я				

Приклад. Відповідно до таблиць стандарту ASCII знайти код для літер «L» і «ж».

Перша літера – це літера англійського алфавіту, тому для пошуку її коду скористаємося табл. 3.6. Рухаючись вниз по стовбцю, знайдемо код, де розташовано символ «L». Це код 7. Рухаючись вправо по рядку, знайдемо код на перетині. Це код 6. Отже, шуканий код літери «L» – 76.

Друга літера – це літера українського алфавіту, тому для пошуку її коду скористаємося табл. 3.7. Аналогічно, рухаючись вниз по стовбцю таблиці, знайдемо код, де розташовано символ «ж». Це код 23. Рухаючись вправо по рядку, знайдемо код на перетині. Це код 0. Отже, шуканий код літери «ж» – 230.

За наведеним у табл. 3.7 фрагментом розширеної частини таблиці ASCII визначають коди для символів кирилиці. Ця система кодування, більш відома як система кодування Windows-1251, була введена компанією Microsoft. Завдяки широкому використанню операційних систем та інших продуктів компанії ця розширена частина таблиці ASCII глибоко закріпилася і є поширеною.

Приклад *Стандарт ASCII за замовчуванням застосовують у мовах програмування високого рівня. Так, при програмуванні мовою С для роботи з текстом використовують тип даних* ***char****, розмірність якого завжди дорівнює байту. Відповідно до табл. 3.6 і 3.7 цей тип даних можна інтерпретувати як ціле число і як символ тексту. При цьому над змінною типу* ***char****, що зберігає код символу, можна виконувати будь-які операції як над цілою змінною, починаючи від порівняння та присвоєння і закінчуючи операціями над окремими розрядами.*

Зазначимо, що застосування стандарту ASCII визначає певні кількісні характеристики інформаційних об’єктів (наприклад, текстових даних), зокрема, їх інформаційний об’єм – розмір, який інформаційний об’єкт займає в пам’яті комп’ютера (або будь-якого електронного пристрою).

Інформаційний об’єм знаходять за формулою

$$I = K \cdot n, \qquad (3.5)$$

де K – кількість символів у тексті, збереженому в комп’ютері; n – розрядність двійкового коду, яким подано символи тексту.

Приклад. Знайти інформаційний об’єм текстового файлу, який містить 12 рядків із 28 символів, представлених у системі кодування ASCII.

Знайдемо кількість символів у файлі

$$K = 12 \cdot 28 = 336.$$

З урахуванням розрядності коду ASCII (n =8) інформаційний об’єм становить

$$I = K \cdot n = 336 \cdot 8 = 2\,688 \text{ бітів, або } 336 \text{ байтів.}$$

Для японської та китайської мов 256 кодів символів стандарту ASCII виявилося недостатньо. У цьому випадку використовують багатобайтові кодування, зокрема, стандартні таблиці DBCS (Double Byte Character Set – двобайтовий набір символів).

DBCS – це кодування, при якому або всі символи (включаючи символи керування), або кожен графічний символ, якого немає в однобайтовому наборі символів (наприклад, ASCII або EBCDIC), закодовано двома байтами. Таким чином, DBCS підтримує національні мови, що містять велику кількість унікальних знаків або символів. Найпоширенішим прикладом цієї системи кодування є розширення національного стандарту GBK, що підтримує спрощену та традиційну китайські мови, кількість символів у якому становить близько 21 000 ієрогліфів [29].

Примітка *Стандарт GBK містить всі символи, що застосовуються в КНР і на Тайвані, де-факто є стандартом традиційної китайської мови для персональних комп'ютерів. У цей час GBK розглядають як проміжну ланку для переходу до Unicode. Тестувальники застосовують його при тестуванні процесів друкування.*

Прагнення розширити кількість мов, що підтримуються, необхідність створення систем кодування, частково сумісних з будь-якою іншою, призвели до того, що на початку 90-х рр. XX ст. поряд зі стандартними символами ASCII існувало безліч різних 8-бітових систем кодувань і постійно з'являлися нові. Унаслідок цього розробники і користувачі комп'ютерних та інформаційних систем постійно стикалися з низкою проблем, зокрема [1, 30]:

1. Проблемою відображення документів у неправильному коді (проблема «кракозябр»), вирішення якої полягало або в обов'язковому зазначенні методу кодування (наприклад, це застосовують при написанні html-документів), або впровадженні єдиного методу кодування для всіх.
2. Проблемою обмеженого набору символів, що вирішується або під'єднанням відповідних шрифтів для відображення документа (наприклад, це використовують дизайнери web-сторінок), або використанням «широкого» кодування.

Примітка *Під'єднання шрифтів спочатку використовувалося в текстових процесорах – комп'ютерних програмах з написання та модифікації документів, компонування макета тексту, його попереднього перегляду тощо. Однак якщо в документі використовувалися шрифти з нестандартним кодуванням, що широко розповсюджено, наприклад, в іграх, де часто застосовують власний шрифтовий «движок», то спроба переведення його в іншу систему закінчувалася перетворенням нестандартних символів на «кракозябри».*

3. Проблемою перетворення символів однієї системи кодування на символи іншої, яку вирішують складанням таблиць перекодування для кожної пари систем кодувань або використанням проміжного подання в третю систему кодування, що містить усі символи для перших систем кодування.
4. Проблемою дублювання шрифтів за умови, що для кожної системи кодування створювали свій шрифт (навіть якщо ця система частково або повністю збігалася за відомим набором символів). Цю проблему вирішували, створюючи «великі» шрифти, з яких за потреби вибирали

необхідні для кодування символи, що, на жаль, потребувало створення єдиного реєстру символів для встановлення відповідності між певними шрифтами та системами кодувань.

Таким чином, було визнано необхідним створення єдиної системи «широкого» кодування. Така система, основана на 16-розрядному кодуванні символів, отримала назву універсальної – Unicode. Стандарт цієї системи кодувань – ISO/IEC 10646 – вперше запропонувала в 1991 р. некомерційна організація «Консорціум Unicode Inc».

Примітка *При формуванні вимог до нової системи кодування розглядалися різні варіанти. Кращою виявилася система з фіксованою довжиною символів, тому що кодування зі змінною довжиною символів, яке, наприклад, дуже поширено в Східній Азії, було занадто складним у використанні. Застосування 32-бітових символів здавалося надто марнотратним, тому використовується 16-бітова довжина символів.*

Стандарті Unicode складається з двох основних частин [30]:

– UCS (Universal Character Set) – універсальний набір символів, який визначає допустимі за стандартом символи і ставить кожному з них відповідний код у вигляді невід'ємного цілого шістнадцяткового числа, що записують з префіксом «U+»;

– UTF (Unicode Transformation Format) – спосіб кодування, що визначає способи перетворення кодів символів на машинне подання з використанням змінної кількості байтів для компактнішого збереження та передавання символів Unicode.

Перша версія Unicode являла собою систему кодування з фіксованим 16-бітовим розміром символу, що зумовило позначення чотирма шістнадцятковими цифрами (наприклад, U+04F0). Вона визначала $2^{16} = 65\,536$ різних елементів кодового простору та забезпечувала унікальні коди символів, необхідних у повсякденному побуті 25 різних писемностей.

Еволюція Unicode сприяла тому, що коди символів стали розглядати не як 16-бітові значення, а як абстрактні числа, що мають декілька форм подання в комп'ютері (наприклад, визначені системами кодування UTF-8, UTF-16, UTF-32 тощо). Однак упровадження за замовчуванням у деяких комп'ютерних системах (наприклад, Windows NT) фіксованих 16-бітових символів зумовило те, що для сумісності з ними було розроблено систему UTF-16, яка найважливіші символи кодує в межах перших 65 536 позицій, а решту простору використовує для кодування «додаткових символів» (систем письма «мертвих» мов, китайських ієрогліфів, які дуже рідко застосовують, математичних, музичних символів тощо).

Система UTF-16 – система кодування символів Unicode, де кожному символу ставлять у відповідність одне або два 16-розрядних числа з діапазону 0x000000 … 0x10FFFF, що значно розширює обсяг кодового простору, тобто [31]

$$V_{unicode} = (2^{16} - 2048) + 2^{10} * 2^{10} = (2^{16} - 2048) + 2^{20} = 1112064,$$

де (2^{16}-2048) – обсяг кодового простору для символів Unicode, які подають

16-розрядним числом з діапазону 0x0000 … 0xD7FF або 0xE000 … 0xFFFF; 2^{20} – обсяг кодового простору для символів Unicode, що подають «сурогатними парами» – двома 16-розрядними числами з діапазону 0xD800 … 0xDFFF.

Символи діапазону 10000_{16} … $10FFFF_{16}$ (більше 16 бітів) кодують за схемою:

1. Від коду символу віднімають 10000_{16}, отримують значення від 0000_{16} до $FFFFF_{16}$, що поміщується в 20-бітову розрядну сітку.

2. Старші 10 бітів (числа з діапазону 0000_{16} … $03FF_{16}$) додають до $D800_{16}$, результат записують у перший байт сурогатної пари (перше слово) з діапазону $D800_{16}$ … $DBFF_{16}$.

3. Молодші 10 бітів (числа з діапазону 0000_{16} … $03FF_{16}$) додають до $DC00_{16}$, результат записують у другий байт сурогатної пари (друге слово) з діапазону $DC00_{16}$ … $DFFF_{16}$.

Кодовий простір $V_{unicode}$ поділено на 17 площин, де під площиною розуміють безперервний діапазон із 65 536 (2^{16}) кодових позицій. Діапазон кодових позицій та їх взаємозв'язок з площинами наведено в табл. 3.8.

Таблиця 3.8

Площини UTF-16

Номер площини	Діапазон кодових позицій	Назва площини
0	0x0000 – 0xFFFF	Основна багатомовна (англ. Basic Multilingual Plane, BMP)
1	0x10000 – 0x1FFFF	Додаткова багатомовна (англ. Supplementary Multilingual Plane, SMP)
2	0x20000 – 0x2FFFF	Додаткова ідеографічна (англ. Supplementary Ideographic Plane, SIP)
3	0x30000 – 0x3FFFF	Третинна ідеографічна (англ. Tertiary Ideographic Plane, TIP)
4 –13	0x40000 – 0xDFFFF	Не використовують
14	0xE0000 – 0xEFFFF	Спеціалізована додаткова (англ. Supplementary Special-purpose Plane, SSP)
15	0xF0000 – 0xFFFFF	Додаткова площина для окремого використання – A (англ. Supplementary Private Use Area - A, SPUA - A)
16	0x100000 – 0x10FFFF	Додаткова площина для окремого використання – B (англ. Supplementary Private Use Area - B, SPUA - B)

На сучасному етапі спостерігаємо переведення документів і програмних засобів на цю універсальну систему кодування, тому що, як стверджує консорціум Unicode: «…це унікальний код для будь-якого символу незалежно від платформи, від програми, від мови» [30]. Це стає можливим завдяки таким принципам:

– гарантії стабільності, що забезпечується тим, що якщо символ з'явився у Unicode, то він не зникає, а стає надмножиною старого символу;

– динамічної компоновки, коли складні символи динамічно складаються під час друкування тексту (рис. 3.2);

– логічного порядку, тобто порядок символів приблизно збігається з порядком читання та набору;

– перетворюваності, що передбачає повторення Unicode принципів кодування символів з вихідної системи кодування;

– простого тексту, тобто Unicode кодує текст без оформлення;

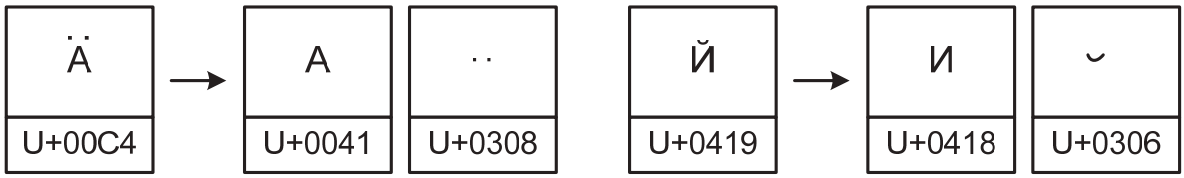


Рис. 3.2. Компонування символу шляхом поєднання базових і комбінованих символів

– семантики, який передбачає, що властивості символів задають формально;

– кодування одиниці смислу, тобто символ Unicode – це одиниця смислу, а не його зображення;

– універсальності, тобто Unicode створено для різних людей різних мов і професій, що працюють в різних сферах, для сучасних мов та історичних текстів;

– уніфікації, який реалізується шляхом недублювання символів, наприклад, англійська літера «вай», французька «ігрек» і німецька «іпсилон» – та сама кодова позиція Y, хоча схожі букви різних писемностей кодують різними кодовими позиціями, наприклад, «A» латиниці кодують U+0041, а «А» кирилиці – U+00C0.

– ефективності через подання кодів символів послідовністю чисел від 0 до 10FFFF16, що дає змогу використовувати таблиці пошуку.

Незважаючи на значний обсяг кодового простору, що дає змогу подавати текст будь-якою мовою, перехід на систему Unicode довгий час стримувався через обмеження ресурсів засобів обчислювальної техніки в глобальних інформаційних системах.

Сьогодні більшість сучасних операційних систем деякою мірою забезпечує підтримку Unicode, а однією з перших успішних комерційних реалізацій Unicode стало створення середовища програмування Java, де принципово відмовилися від 8-бітового подання символів на користь 16-бітового. Однак при цьому слід мати на увазі таке:

1. Файли неанглійського тексту в Unicode займають завжди більше місця, тому що на відміну від національних систем кодування, один символ тут кодують послідовністю байтів.

2. Файл шрифту, необхідного для відображення всіх символів таблиці Unicode, займає більше місця в пам'яті і потребує більших обчислювальних ресурсів, ніж файл шрифту однієї національної мови користувача.

Примітка *Зі збільшенням потужності комп'ютерних систем і здешевленням пам'яті та дискового простору ця проблема стає все менш суттєвою, але вона залишається актуальною для портативних пристроїв.*

3. Досі не все прикладне програмне забезпечення підтримує коректну роботу Unicode, хоча його підтримка реалізована в найпоширеніших операційних системах.

Приклад *Не завжди обробляються мітки порядку байтів при використанні «сурогатних пар» і погано підтримуються діакритичні символи.*

4. Продуктивність усіх програм оброблення рядків (у тому числі під час сортування в базах даних) знижується при використанні Unicode порівняно з ASCII.

Приклад. Текстове повідомлення з 20 символів, написане українською мовою, подано у коді Unicode. Автоматичний пристрій перекодував його у код ASCII. Як змінився розмір вихідного повідомлення?

За формулою (3.5) знайдемо розмір вихідного повідомлення, поданого:

– у Unicode (n =16):

$$I = K \cdot n = 20 \cdot 16 = 320 \text{ бітів};$$

– у коді ASCII (n =8):

$$I = K \cdot n = 20 \cdot 8 = 160 \text{ бітів}.$$

Отже, розмір вихідного повідомлення зменшився у 2 рази або на 160 бітів.

3. Подання числової інформації в комп'ютерах

Спосіб отримання числової інформації визначає її клас, який, у свою чергу, визначає правила запису числових бітів у форматі [1].

Числову двійкову інформацію можна подати:

– у форматі з фіксованою точкою (або в природній формі), який використовують для зображення цілих чисел;

– форматі з рухомою точкою (або в нормальній формі), який використовують для зображення дійсних чисел.

Число у форматі з фіксованою точкою зображується як послідовність цифр з постійним для них місцем розташування точки, що відокремлює цілу частину числа від дробової.

Положення точки відносно старшого розряду зазвичай однаково для всіх чисел і встановлено раз і назавжди – після останньої цифри числа або перед першою цифрою числа (див. формулу (3.1)). Така фіксація точки дає змогу опускати її в зображенні числа [21].

Якщо при поданні n-розрядного двійкового числа з фіксованою точкою точка знаходиться після його останньої цифри, то діапазон зміни його значень буде таким [1, 21]:

$$0 \le |A_2| \le 2^n - 1.$$

При такому поданні внаслідок виконання операцій над числами з фіксованою точкою можна отримати результат, що виходить за допустимий діапазон. Подальші обчислення втрачають сенс, а таке явище називають переповненням розрядної сітки. Тому в сучасних

комп'ютерних технологіях таку форму подання чисел використовують як допоміжну і лише для цілих чисел, створюючи їх через спеціальні функції (табл. 3.9).

У ГІС-додатках при зберіганні цілі числа відповідно до формату можуть займати поля довжиною у пів слова (1 байт), слово (2 байти) або подвійне слово (4 байти). При цьому використовують два способи подання таких чисел [32]: беззнакове – для додатних чисел і знакове – для від'ємних. Усе це задає відповідна функція, яка обмежує можливий діапазон числа.

Таблиця 3.9

Функції Scilab для створення цілих чисел з урахуванням формату та способу подання

Знакове подання			Беззнакове подання		
Ім'я функції	Формат числа	Діапазон	Ім'я функції	Формат числа	Діапазон
int8()	8-бітове число	$-2^7 – (2^7-1)$	**uint**8()	8-бітове число	$0 – (2^8-1)$
int16()	16-бітове число	$-2^{15} – (2^{15}-1)$	**uint**16()	16-бітове число	$0 – (2^{16}-1)$
int32()	32-бітове число	$-2^{31} – (2^{31}-1)$	**uint**32()	32-бітове число	$0 – (2^{32}-1)$

Неважко помітити, що якщо числова величина є додатною, то вигідніше розглядати її як беззнакову.

Для зберігання цілого числа залежно від формату в пам'яті комп'ютера виділяють відповідну кількість розрядів, куди, починаючи з молодшого розряду, поміщують числові дані. При цьому в знаковому поданні додатні числа мають нульове знакове значення в старшому біті числа, від'ємні – одиничне [21].

Приклад. Навести структурний запис чисел $A = +17_{10}$, $B = -193_{10}$, використовуючи поля необхідної довжини.

Переведемо числа A і B у двійкову систему числення:

$$A = +17_{10} = +10001_2 \text{ та } B = -193_{10} = -11000001_2.$$

Число A – додатне, п'ятирозрядне, для його подання використаємо формат пів слова (довжиною 1 байт), доповнюючи незначущими нулями відповідні розряди:

	Код знака	Абсолютне значення числа							
Номер розряду	8	7	6	5	4	3	2	1	0
Цифри числа	0	0	0	0	1	0	0	0	1

Число B – від'ємне, восьмирозрядне, тому його подання у форматі пів слова (довжиною 1 байт) є неможливим (не вистачає місця для коду знака). Використаємо формат слова (довжиною 2 байти):

	Код знака	Абсолютне значення числа													
Номер розряду	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Цифри числа	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1

Використання двійково-десяткового кодування чисел з фіксованою точкою дає змогу подавати їх полями змінної довжини в розпакованому або упакованому форматі.

У розпакованому форматі відповідно до стандарту ASCII для подання кожної десяткової цифри відводиться байт. При цьому старший півбайт називають зоною і заповнюють кодом 0011. У молодших півбайтах звичайним чином кодують десяткові цифри. Зона наймолодшого байта використовується для кодування знака числа, значення якого вибирають із множини заборонених комбінацій, і залежить від системи кодування [28]:

– у системі EBCDIC, яку використовує фірма IBM, знак «+» позначено через 1010, а знак «-» – через 1011;

– у системі ASCII знак «+» позначено через 1100, а знак «-» – через 1101.

Структура полів розпакованого формату має вигляд, зображений на рис. 3.3.

Зона	Цифра	Зона	Цифра	…	Зона	Цифра	Знак	Цифра

Рис. 3.3. Розпакований формат двійково-десяткових чисел

Розпакований формат у комп'ютерах використовують під час введення-виведення інформації, а також при виконанні операцій множення та ділення десяткових двійково-кодованих чисел.

В упакованому форматі для кожної десяткової цифри відводиться по чотири двійкових розряди (півбайт). При цьому знак числа кодується в крайньому правому півбайті числа. Структуру полів в упакованому форматі показано на рис. 3.4.

Цифра	Цифра	…	Цифра	Цифра	Знак

Рис. 3.4. Упакований формат двійково-десяткових чисел

Упакований формат у комп’ютері використовують під час виконання операцій додавання або віднімання двійково-десяткових чисел.

Неважко помітити, що застосування упакованого формату двійково-десяткових чисел дає змогу зменшити кількість бітів, що використовують для подання цифри, з 7 до 4, а отже, прискорити процес маніпуляції з десятковими цифрами.

Для зручності відображення дійсних чисел використовують форму запису чисел з порядком основи системи числення, наприклад, десяткове число 17,5 можна подати так [1, 4]:

$$17.5\cdot 10^{0} = 1.75\cdot 10^{1} = 0.175\cdot 10^{2} = \ldots \text{ або } 175.0\cdot 10^{-1} = 1750.0\cdot 10^{-2} = \ldots,$$

або в нормалізованому експоненціальному вигляді:

$$1.75\cdot 10^{+1} = 1.75\cdot \exp_{10} +1.$$

У такому запису можна відокремити дві частини: мантису $m_a = 1.75$ та експоненту $\exp_{10} = +1$, подану основою системи числення (в цьому випадку – 10) і порядком (в цьому випадку – +1).

Подібний спосіб подання дійсних чисел називають формою подання з рухомою точкою, а числа – числами у нормальній формі.

Основою для подання чисел у нормальній формі є стандарт IEEE 754-1985.

Примітка *Стандарт IEEE 754-1985 повністю трансформовано до стандарту IEEE 754-2008, доповнено вимогами ISO та є основним у комп'ютерній техніці та програмуванні.*

Це найпоширеніший стандарт для здійснення обчислень з числами з рухомою точкою, який використовують мікропроцесори та логічні пристрої, а також програмні засоби. Кількість операцій з рухомою точкою за секунду (Floating Operations Per Second, Flops, у наш час частіше говорять про ГFlops за секунду, наприклад, процесор Intel Core i9-9900k виконує 460 ГFlops/с, а AMD EPYC 7H12 – навіть 4,2 TFlops/с), які може виконувати процесор, є однією з важливих його позасистемних характеристик, що використовують для вимірювання продуктивності комп'ютерів [33].

У комп'ютері використовується двійкова система числення, в нормальній формі число вважають зведеним до вигляду

$$A_2 = \pm m_a \cdot 2^{\pm p_a} = \pm m_a \cdot \exp_2(\pm p_a), \qquad (3.6)$$

де m_a – мантиса числа, подана в двійковій системі числення ($1 < |m_a| < 10$); p_a – порядок, тобто ціле число, яке подано в двійковій системі числення.

Приклад. Записати двійкове число $A_2 = 101_2$ в нормальній формі.

З використанням формули (3.6) вихідне число запишемо так:

$$A_2 = +101 = +1.01 \cdot \exp_2 +10,$$

де двійкова мантиса числа A_2 – $m_a = +1.01_2$, дійсний порядок числа A_2 – $p_a = 10_2$.

Зазначимо, що робоча система числення має невід'ємну базу, тому мантиса і порядок мають знак у знаковому розряді. Однак додатні та від'ємні значення порядку ускладнюють оброблення чисел. Тому, щоб не зберігати його знак, застосовують зміщений порядок [34]:

$$p_a' = p_a + (2^{b-1} - 1), \qquad (3.7)$$

де p_a – дійсний порядок числа; $(2^{b-1} - 1)$ – зміщення порядку; b – кількість бітів, відведених під запис порядку числа.

З урахуванням (3.7) зміщений порядок можна вважати додатним беззнаковим цілим числом.

Стандарт IEEE 754 регламентує чотири формати подання чисел з рухомою точкою [4, 34], серед яких найбільш поширеними є (табл. 3.10):

- з одинарною точністю, або короткий дійсний формат (single-precision), – число довжиною 32 біти;
- подвійною точністю, або довгий дійсний формат (double-precision), – 64 біти;
- подвійною розширеною точністю, або подовжений дійсний формат (double-extended precision), – 80 бітів.

Зазвичай програмістам дається можливість вибрати з числових

форматів той вид запису, який є найбільш прийнятним для вирішення конкретного завдання. Але якщо додаткових даних немає, то найчастіше тип даних – double (навіть у разі, коли змінна – ціле число).

Примітка *Основного застосування в техніці та програмуванні набули формати довжиною 32 і 64 біти. Наприклад, у VB використовують типи даних single (32 біти) і double (64 біти). У Сі використовують float (32 біти) і double. У Scilab, якщо немає додаткових указівок, всі числові змінні – дійсні з подвійною точністю, тобто мають тип double. Аналогічно в мові JavaScript цілі та дійсні числа не розрізняються, внутрішньо їх подають числами у восьмибайтовому форматі з рухомою точкою. У Паскалі використовують типи single, double і extended (80 бітів).*

Таблиця 3.10

Можливі формати дійсних чисел

Назва формату	Довжина числа W, біти	Кількість бітів під зміщений порядок b	Зміщення порядку $(2^{b-1}-1)$	Кількість бітів під мантису n
Короткий дійсний	32	8	127	23
Довгий дійсний	64	11	1023	52
Подовжений дійсний	80	15	16383	64

З урахуванням можливих форматів чисел дійсний запис числа (3.6) відповідно до стандарту IEEE 754 має такий вигляд (рис. 3.5) [4]:

$$A_2 = \pm m_a \cdot 2^{p_a'} = (-1)^S 1.F_i \cdot \exp_2 p_a', \text{ при } i = \overline{1,n}, \qquad (3.8)$$

де S – біт знака (якщо $S=0$, то число є додатним; якщо $S=1$ – від'ємним); p_a' – зміщений порядок, розрахований за формулою (3.7); F_i – двійкові біти залишку від мантиси; n – кількість бітів, що виділяють під мантису дійсного числа у різних форматах (див. табл. 3.10); $F_0 = 1$ – прихований (неявний) біт, який у короткому і довгому дійсних форматах при передаванні чисел та їх зберіганні не фігурує (тільки під час роботи з дійсними числами у форматі double-extended цей біт має явну форму); b – кількість бітів, відведених під зміщений порядок для зображення дійсних чисел різних форматів.

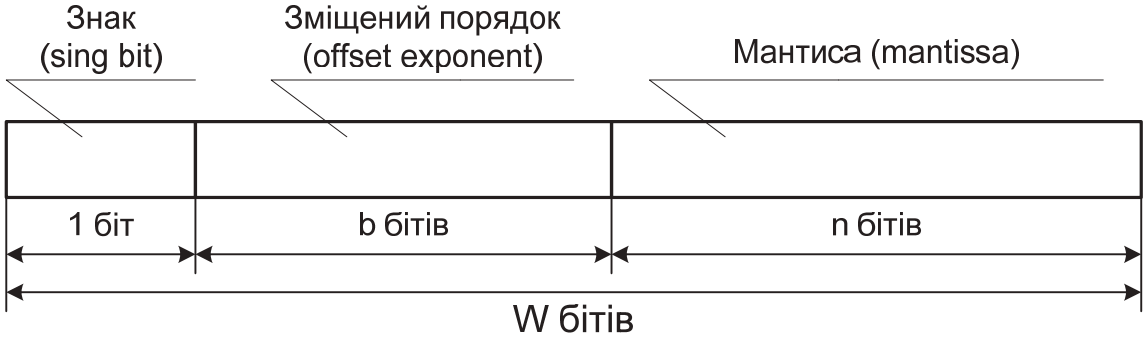


Рис. 3.5. Структурний запис чисел відповідно до стандарту IEEE 754

Приклад. Навести структурний запис десяткового числа A_{10} = -247,375 у короткому дійсному форматі (формат single).

Переведемо вихідне число у двійкову систему числення, пам'ятаючи, що переведення цілої та дробової частин числа здійснюється за різними правилами:

$$-247{,}375_{10} = -11110111{,}011_2.$$

За виразом (3.6) подамо вихідне число в нормальній формі:

$$A_2 = -1.1110111011 \cdot \exp_2 111.$$

За формулою (3.7) з урахуванням даних табл. 3.10 знайдемо зміщений порядок для короткого дійсного формату:

$$p_a' = p_a + (2^{b-1} - 1) = (7 + 127)_{10} = 10000110_2.$$

Ураховуючи наявність прихованого біта в числі, який у короткому форматі не фігурує, структурний запис вихідного числа матиме такий вигляд:

Знак числа	Зміщений порядок								Залишок від мантиси												
Номер розряду	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	...	0
Цифри числа	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**...**	**0**
		C			**3**					**7**				**7**				**6**				**000**

Зазначимо, що дійсні числа не дуже зручно зображувати в двійковій системі числення, тому в стандарті IEEE 754 використовують шістнадцяткове подання. Для цього одержаний структурний запис поділимо на тетради і, використовуючи дані табл. 3.4, переведемо у шіснадцяткову систему числення.

Отже, вихідне число $-247{,}375_{10}$ у короткому дійсному форматі буде A_{single} = C3776000.

Відповідно до стандарту IEE 754 для ознайомлення з особливостями формування структурного запису числа в довгому дійсному форматі (формат double) зручно використовувати функцію Scilab **bitstring()**, що показує двійкову послідовність структурного запису числа.

Приклад. З використанням Scilab сформувати структурний запис десяткового числа A_{10} = -247,375 у довгому дійсному форматі (формат double). Визначити в цьому запису основні складові частини відповідно до рис. 3.5.

За правилами Scilabe (див. додаток) сформуємо файл-сценарій для отримання структурного запису числа A (рис. 3.6, а).

Результат виконання файлу-сценарію показано на рис. 3.6, б.

Приклад. Змінну A подано у форматі з рухомою точкою в шістнадцятковій системі числення: $A = C2F20000_{16}$. Тип змінної – single. Знайти десяткове значення числа A.

Користуючись правилами переведення з шістнадцяткової системи числення в двійкову, перетворимо вихідне подання числа на двійкове:

A_{single} = 1100 0010 1111 0010 0000 0000 0000 0000.

У цьому запису виділимо знак, порядок, мантису.

```
a = bitstring(-247.375); // Введення вихідних даних прикладу
disp('Число у довгому дійсному форматі ', a); // Виведення на екран структурного запису
mprintf("знаковий біт = %s\n",part(a,1)); // Виведення частини структурного запису
mprintf("зміщений порядок = %s\n",part(a,2:12)); // Виведення частини структурного запису з діапазону, що відповідає порядку числа
b=bin2dec(part(a,2:12)); // Переведення порядку числа в десяткову систему числення
printf('зміщений порядок у десятковій системі числення=%f\n', b) // Виведення десяткового значення зміщеного порядку числа
c=b-1023; // Визначення реального порядку числа
printf('реальний порядок у десятковій системі числення=%f\n', c);
mprintf("залишок від мантиси = %s\n",part(a,13:$)); // Виведення частини структурного запису з діапазону, що відповідає залишку від мантиси числа.
```

а

```
--> exec('C:\Users\Admin\Documents\Приклад_4.sce', -1)

  "Число у довгому дійсному форматі "

  "11000000011011101110110000000000000000000000000000000000000000000"
знаковий біт = 1
зміщений порядок = 10000000110
зміщений порядок у десятковій системі числення=1030.000000
реальний порядок у десятковій системі числення=7.000000
залишок від мантиси = 11101110110000000000000000000000000000000000000000000
```

б

Рис. 3.6. Формування структурного запису числа А у довгому дійсному форматі з використанням Scilab: а – лістинг програми; б – структурний запис числа А та його основні частини

Знак числа – старший біт – дорівнює одиниці, отже, число від'ємне.

Порядок числа пам'яті комп'ютера зберігається зі зміщенням. Знайдемо його:

$$p_a^{'} = 10000101_2 = 133_{10}.$$

Віднімаючи від зміщеного порядку число 127_{10} (для змінних типу single), отримаємо значення реального порядку – 6_{10}. Отже, знак точки в мантисі слід змістити вправо на шість позицій.

Для змінних типу single перший біт мантиси – прихований, тому реальна мантиса має вигляд $m_a = 1.111001_2$.

Поєднуємо знак, мантису та порядок:

$$A_2 = -1.111001 \cdot \exp_2 110 = -1111001_2 = -121_{10}.$$

Фахівці зазначають, що використання чисел у стандарті IEEE 754 в деяких випадках може призвести до низки помилок комп'ютерних обчислень, зокрема [34]:

1. До помилок, пов’язаних з точністю подання дійсних чисел в форматі IEEE 754.

При поданні чисел, розмірність яких більша за відведену розрядну сітку, стандарт IEEE 754 за замовчуванням передбачає округлення до найближчого цілого числа. При цьому в стандарті IEEE 754 передбачено чотири способи округлення (табл. 3.11): округлення до найближчого цілого; округлення до нуля; округлення до +∞; округлення до -∞.

Таблиця 3.11

Приклад округлення чисел до десятих

Вихідне число	Спосіб округлення			
	до найближчого цілого	до нуля	до +∞	до -∞
1,33	1,3	1,3	1,4	1,3
-1,33	-1,3	-1,3	-1,3	-1,4
1,37	1,4	1,3	1,4	1,3
-1,37	-1,4	-1,3	-1,3	-1,4
1,35	1,4	1,3	1,4	1,3
-1,35	-1,4	-1,3	-1,3	-1,4

Примітка. Для апаратної реалізації найпростішим є спосіб округлення до нуля, який відкидає незначні розряди числа.

Розглянемо такий приклад.

З використанням стандарту IEEE 754 у форматі single відняти два числа: 123456789_{10} і 123456788_{10}.

Якщо приклад прорахувати вручну, то відповідь буде дорівнювати 1, але використання стандарту IEEE 754 дає результат 8. Проаналізуємо, чому відповідь виходить неправильною.

Зазначимо, що вихідні числа не укладаються в розрядну сітку в 32 біти, передбачену для подання чисел у форматі single. Тому число 123456789 у форматі single з урахуванням правил округлення запишемо як $4CEB79A3_{16}$, що в десятковій системи числення відповідає числу 123456792_{10}. Отже, абсолютна помилка подання першого числа становить +3.

Аналогічно число 123456788 у форматі single запишемо як $4CEB79A2_{16}$, що в десятковій системи числення відповідає числу 123456784_{10}. Таким чином, абсолютна помилка подання другого числа дорівнює -4.

У результаті абсолютна різниця між вихідними дійсними числами у форматі single сягає 8.

Таким чином, при поданні дійсних чисел, формат яких сформовано відповідно до стандарту IEEE 754, потрібно ураховувати помилки, що призводять до зниження точності обчислень в операціях, де абсолютне значення результату значно менше за будь-яке значення вихідних змінних (як у розглянутому прикладі).

2. До помилок, пов’язаних з неправильним застосуванням типів даних, що пояснюється тим, що подання одного й того числа у форматах single і double зазвичай призводить до різних результатів, тобто числа, отриманні внаслідок переведення, не дорівнюють один одному.

Пояснимо це на такому прикладі.

Необхідно подати число $123456789{,}123456789_{10}$ у форматах single і double та знайти різницю.

Як у попередньому випадку, вихідне число не укладається в розрядну сітку в 32 біти, передбачену для подання чисел у форматі single. Використовуючи округлення, це число в форматі single запишемо так: $4CEB79A3_{16}$, що в десятковій системі числення відповідає числу $123456792{,}0_{10}$.

У форматі double це число подамо як $419D6F34547E6B75_{16}$, що в десятковій системі числення відповідає числу $123456789{,}123456791000000000000000_{10}$.

Різниця між числами в форматах single і double становить 2,8765432109.

Зазначимо, що переважна більшість чисел, поданих відповідно до стандарту IEEE 754, має стабільну невелику відносну похибку. Наприклад, максимально можлива відносна похибка для чисел у форматі single становить 2^{-23}. Тому змінні та проміжні результати комп'ютерних обчислень слід подавати в одному форматі.

Вимоги зведення даних до одного типу формату викладено в багатьох стандартах на мови програмування високого рівня. Ці вимоги також стосуються й проміжних результатів обчислень.

Приклад *У стандарті ISO/IEC 9899:1999 на мову програмування C найпоширенішим є формат double, що забезпечує необхідну точність обчислень арифметичних виразів. Тому будь-яку змінну формату float перед застосуванням автоматично перетворюють на змінну формату double. Крім того, якщо в операції є одна змінна типу double, а інша – цілим числом, то останню також зводять до формату double.*

Також зазначимо, що якщо розглядати дійсні числа як вимірювальну інформацію, то такого типу дані можна отримати за допомогою приладів, точність яких не вища від класу 0.1%, що відповідає 4 – 5 розрядам після точки. Тому більшість мантис IEEE754-чисел, особливо у форматах подвійної та розширеної точності, є незмістовними.

Тестові запитання та завдання для самоперевірки

1. Сформуйте базу для таких систем числення:

а) для двійкової;

б) для вісімкової;

в) для десяткової;

г) для шіснадцяткової.

2. Продовжіть речення: «У позиційній системі числення …

а) використовуються лише арабські цифри;
б) базу складають тільки арабські цифри;
в) кількісне значення цифри залежить від її позиції в числі;
г) цифра множиться на основу системи числення.

3. Яку систему числення найчастіше застосовують як еквівалент двійкової під час налаштування обчислювальних систем:

а) десяткову;
б) вісімкову;
в) шіснадцяткову?

4. Установіть відповідність «характеристика – код» (наприклад, а – а, б – б та ін.):

Характеристика	Код
а) це система кодування для країн лише з алфавітною писемністю	а) scan-код
б) належить до міжнародної системи кодування символів комп'ютерами та іншими електронними пристроями	б) віртуальний код клавіші
в) це код, за допомогою якого драйвер клавіатури розпізнає, яку з клавіш натискали	в) ASCII
г) це двійкове число, що ідентифікує символ	г) Unicode

5. Фактично у 1 МБ:

а) 1 000 000 бітів;
б) 1024 байтів;
в) 1 000 000 байтів;
г) 1024 КБ.

6. Відомо, що текст складається з символів – букв, цифр, розділових знаків і т. д. Як комп'ютер їх розрізняє:

а) за їх сигналами;
б) за їх двійковим кодом;
в) за їх зображенням?

7. Текст, поданий у стандарті ASCII, займає 10 КБ пам'яті комп'ютера. Скільки символів міститься у цьому тексті:

а) 2500;
б) 625;
в) 1250;
г) 312 КБ?

8. Подайте числа $A = 3A6_{16}$ та $B = 215_8$ у вигляді суми відповідно до виразу (3.2).

9. З використанням виразу (3.2) визначте десятковий запис чисел $A = 101011_2$, $B = 137_8$ та $C = 2C5_{16}$. Перевірте правильність одержаного результату, застосовуючи функції Scilab **bin2dec**(A), **oct2dec**(B) і **hex2dec**(C).

10. Компанія-виробник заявляє, що накопичувач має місткість 320 ГБ. Який обсяг даних можна зберігати на цьому накопичувачі?

11. Подайте в двійковій системі числення числа $A = 15{,}5_{10}$, $B = 0{,}375_{10}$ і $C = 4{,}25_{10}$.

12. Запишіть константу A = 77_{10} у «машинних» системах числення.

13. За таблицями стандарту ASCII закодуйте своє прізвище, сформуйте вектор десяткових кодів символів V=[v_1 v_2 … v_m]. Використовуючи стандарт ASCII з урахуванням правила формування віртуального коду, сформуйте відповідний код прізвища за допомогою функції Scilab **dec2bin**(V,n), де n – необхідна розрядність віртуального коду. З використанням функції Scilab **dec2hex**(V) перекодуйте своє прізвище у Unicode.

14. Цитату Ліни Костенко: «Люди, як правило, бачать світ у діапазоні своїх проблем» подано у коді Unicode. Оцініть її інформаційний об'єм.

15. Наведіть структурний запис цілих чисел A = -134_{10} і B = $+261_{10}$, для зручності використовуючи функцію Scilab **dec2bin**(**abs**(x)), де **abs**(x) – абсолютне значення числа, тому що функцію **dec2bin**() використовують для беззнакових чисел.

16. Подайте числа A = -134_{10} і B = $+261_{10}$ у двійково-десятковому коді. З використанням форматів змінної довжини наведіть їх структурний запис за умови, що коди знаків вибрано відповідно до стандарту ASCII. Оцініть ефективність застосування упакованого формату порівняно:

а) з форматом цілих чисел з фіксованою точкою;

б) з розпакованим форматом двійково-десяткових чисел.

17. Подайте числа A = -3,75 і B = 5,0 у двійковому вигляді в природній і нормальній формах.

18. Сформуйте структурний запис десяткового числа A = -3,75 у короткому дійсному форматі.

19. Знайдіть десятковий запис змінної $A_{single} = 40F30000_{16}$, поданої в короткому дійсному форматі.

20. Сформуйте структурний запис десяткового числа A = -14,625 у довгому дійсному форматі (формат double) із застосуванням функції Scilab **bitstring**(x). Визначте в цьому запису основні структурні елементи відповідно до стандарту IEEE 754.

Лекція 4. АРИФМЕТИКО-ЛОГІЧНІ ОСНОВИ КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

План

1. Машинні коди.
2. Арифметичні основи комп'ютерних технологій.
3. Логічні основи комп'ютерних технологій.

1. Машинні коди

Уже зазначалося, що під час виконання операцій над числовою інформацією комп'ютери використовують знакове подання чисел, тобто крім двійкового абсолютного значення числа вказують його знак (знак «+» кодується двійковим нулем, знак «-» – двійковою одиницею). Таке подання називають машинним кодом [35].

Машинний код найчастіше розглядають як примітивну мову програмування або найнижчий рівень подання комп'ютерних програм [36].

Усі арифметичні операції виконують над числами, поданими машинними кодами, що дає змогу [18, 20]:

– реалізовувати базові операції арифметико-логічного пристрою – додавання та зсув;

– використовувати однакові правила для оброблення знакових і значущих розрядів чисел;

– зводити базові арифметичні операції до операції додавання;

– спростити визначення знака операції.

Розрізняють прямий, зворотний і додатковий коди двійкових чисел [8, 21].

Прямий код двійкового числа $[A_2]_n$ утворюється з абсолютного значення цього числа і коду знака – подвоєного нульового або одиничного значення перед його старшим числовим розрядом, тобто

$$[A_2]_n = \begin{cases} 00 : a_n a_{n-1}...a_1 a_0, a_{-1}...a_{-m}, \ A_2 \geq 0, \\ 11 : a_n a_{n-1}...a_1 a_0, a_{-1}...a_{-m}, \ A_2 \leq 0, \end{cases} \quad (4.1)$$

де $A_2 = a_n a_{n-1}...a_1 a_0, a_{-1}...a_{-m}$ – вихідне двійкове число, яке має n розрядів в цілій частині та m розрядів у дробовій частині.

Примітка *Для зручності у виразі (4.1) і в подальшому при формуванні запису машинних кодів знаковий розряд від значущих будемо відділяти умовною позначкою «:», а як код знака в знакових розрядах будемо використовувати подвійне значення «00» при кодуванні знака «+» і подвійне значення «11» при кодуванні знака «-».*

Машинні операції над прямими кодами двійкових чисел ускладнені через необхідність урахування значень знакових розрядів, зокрема:

– слід окремо обробляти значущі розряди чисел і розряди знака;

– значення розряду знака впливає на алгоритм виконання операції (додавання може замінятися відніманням і навпаки).

Зворотний код додатного двійкового числа $[A_2]_з$ збігається з його прямим кодом.

Зворотний код від'ємного числа містить одиницю в знаковому розряді, а значущі розряди числа замінюються інверсними, тобто нулі замінюються одиницями, а одиниці – нулями:

$$[A_2]_з = \begin{cases} 00 : a_n a_{n-1} \ldots a_1 a_0, a_{-1} \ldots a_{-m}, & A_2 \geq 0, \\ 11 : \bar{a}_n \bar{a}_{n-1} \ldots \bar{a}_1 \bar{a}_0, \bar{a}_{-1} \ldots \bar{a}_{-m}, & A_2 \leq 0, \end{cases} \quad (4.2)$$

де $\bar{a}_i = 1 - a_i$ – інверсні значення a_i.

Код чисел називається зворотним через те, що коди цифр від'ємного числа замінено на інверсні.

Примітка *Особливістю реалізації запам'ятовувальних пристроїв сучасних комп'ютерів є те, що вони одночасно зберігають як пряме, так й інверсне значення розрядів чисел. Ця особливість в деяких випадках є дуже корисною, наприклад, при розробленні алгоритмів реалізації операцій зі зворотним кодом на низькому (мікропрограмному) рівні [36].*

Найважливіші властивості зворотного коду:

– додавання додатного числа з його від'ємним значенням у зворотному коді дає машинну одиницю $МО_з$ = 11:111…11, яка складається з одиниць у знаковому та значущих розрядах;

– у зворотному коді нуль має подвійне значення:

$$[0_2]_з = [+0_2]_з = \underbrace{00 : 0 \ldots 000}_{2+n+m \text{ нулей}} \text{ – зворотний код додатного нуля;}$$

$$[0_2]_з = [-0_2]_з = \underbrace{11 : 1 \ldots 111}_{2+n+m \text{ одиниць}} \text{ – зворотний код від'ємного нуля,}$$

значення якого збігається з машинною одиницею $МО_з$.

Подвійне значення «0» стало причиною того, що в сучасних комп'ютерах найбільш поширеним стає додатковий код.

Додатковий код $[A_2]_д$ додатних чисел збігається з їх прямим кодом. Додатковий код від'ємного числа являє собою результат підсумовування зворотного коду числа з одиницею молодшого розряду, тобто

$$[A_2]_д = \begin{cases} 00 : a_n a_{n-1} \ldots a_1 a_0, a_{-1} \ldots a_{-m}, & A_2 \geq 0, \\ 11 : \bar{a}_n \bar{a}_{n-1} \ldots \bar{a}_1 \bar{a}_0, \bar{a}_{-1} \ldots \bar{a}_{-m} \underset{n+m+s}{+ \; 0 \ldots 01}, & A_2 \leq 0, \end{cases} \quad (4.3)$$

де $\bar{a}_i = 1 - a_i$ – інверсні значення a_i.

Приклад. Сформувати машинні коди чисел A = -7,625 і B = +10.

Переведемо число A у двійкову систему числення:

A_2 = -111.101 і B_2 = +1010.

За правилами формування кодів отримаємо

$$[A_2]_п = 11{:}111.101,\ [A_2]_з = 11{:}000.010,\ [A_2]_д = 11{:}000.011;$$
$$[B_2]_п = [B_2]_з = [B_2]_д = 00{:}1010.$$

У комп'ютерах застосовують швидкий спосіб формування додаткового коду, який реалізується в такій послідовності [36]:

– двійкове число, подане в прямому коді, проглядається від молодшого розряду до старшого;

– доки у коді зустрічаються нулі, їх копіюють до однойменних розрядів результату;

– першу одиницю, що зустрілася, копіюють у відповідний розряд, а кожен наступний біт вихідного числа замінюють на інверсний (0 замінюють на 1, а 1 – на 0).

Приклад. Сформувати додатковий код числа A = -84 двома способами.

Переведемо число у двійкову систему числення:

$$A_2 = -1010100.$$

За правилами формування коду (4.3) отримаємо

$[A_2]_п = 11{:}\ 1010100,\ [A_2]_з = 11{:}0101011,\ [A_2]_д = 11{:}\ 0101100.$

Другий (швидкий) спосіб формування додаткового коду показано на рис. 4.1.

$$[A_2]_п = 11{:}\underbrace{1010}_{\text{Інвертується}}\ \underbrace{100}_{\text{Зберігається}}$$

$$[A_2]_д = 11{:}0101100.$$

Рис. 4.1. Швидкий спосіб формування додаткового коду

Як видно, результати, отримані двома способами, збігаються.

Важливіші властивості додаткового коду числа:

– додавання у додатковому коді додатного числа з його від'ємним значенням дає так звану машинну одиницю додаткового коду $МО_д = 10{:}000\ldots00$, тобто число 10 (два) у знаковому розряді числа;

– нуль у додатковому коді має лише одне значення, а саме

$$[0]_д = 000...000;$$

– для від'ємних чисел додатковий код є доповненням прямого коду до машинної одиниці $МО_д$ (звідси його назва).

2. Арифметичні основи комп'ютерних технологій

Будь-який комп'ютер, від найпримітивнішого до суперпотужного, має у своїй системі команд команди для виконання арифметичних дій. Працюючи з комп'ютером з використанням мов високого рівня, користувач сприймає можливість проведення розрахунків як щось належне. Але компілятор будь-якої дуже розвиненої мови програмування перетворює всі

високорівневі дії на двійкову послідовність машинних команд. Зазвичай рідко хто програмує розрахункові задачі в машинному коді, але навіть у системних програмах часто потрібне проведення невеликих обчислень. Тому важливо розібратися із цією групою команд, до того ж вона є дуже компактною та ненадмірною [36].

До початку виконання арифметичної дії операнди операції розміщуються у відповідні регістри арифметико-логічного пристрою (див. рис. 2.3).

Додавання двійкових чисел здійснюється послідовно, порозрядно відповідно до табл. 4.1 [17].

Числа, що підлягають додаванню або відніманню, зазвичай подають у зворотному або додатковому коді. Пов'язано це з тим, що в безпосередньому вигляді операція віднімання в комп'ютерах не здійснюється, вона зводиться до операції додавання [4, 37].

Таблиця 4.1

Правила додавання двох двійкових чисел
A і B з урахуванням перенесення

Значення двійкових чисел			Підсумок S_i	Перенесення в наступний розряд C_i
A	B	C_{i-1}		
0	0	0	00	0
0	0	1	01	0
0	1	0	01	0
0	1	1	10	1
1	0	0	01	0
1	0	1	10	1
1	1	0	10	1
1	1	1	11	1

Наприклад, нехай задано $A \geq 0$ та $B \geq 0$. Операція алгебраїчного додавання виконується відповідно до виразів табл. 4.2 [8, 37].

При додаванні чисел необхідно дотримуватися певних правил.

Таблиця 4.2

Перетворення кодів при алгебраїчному додаванні

Необхідна операція	Необхідне перетворення
$A + B$	$A + B$
$A - B$	$A + (-B)$
$-A + B$	$(-A) + B$
$-A - B$	$(-A) + (-B)$

Примітка. Дужки в наведених виразах указують на заміну операції віднімання операцією додавання з додатковим кодом відповідного числа.

Правило 1. Доданки повинні мати однакову кількість розрядів. Для вирівнювання розрядної сітки доданків можна дописувати незначущі нулі зліва до цілої частини числа і справа – до дробової частини числа.

Правило 2. Знакові розряди чисел беруть участь в операції додавання так само, як і значущі.

Правило 3. Необхідні перетворення кодів (з прямого на зворотний і додатковий) виконуються зі зміною знаків чисел. У разі перетворень приписані незначущі нулі змінюють своє значення за загальним правилом.

Правило 4. При утворенні одиниці перенесення зі старшого знакового розряду, у разі використання зворотного коду, ця одиниця додається до молодшого числового розряду. При використанні додаткового коду одиниця перенесення втрачається. Знак результату формується автоматично, результат наводиться в тому коді, в якому подано вихідні складові.

Приклад. Додати два числа: $A_{10} = +16$ і $B_{10} = -7$.

Запишемо вихідні числа у двійковому вигляді:

$$A_2 = +10000 \text{ і } B_2 = -111.$$

Видно, що ці числа мають різну розрядність, тому, користуючись правилом 1, зробимо вирівнювання їхніх розрядних сіток:

$$[A_2]_п = 00: 10000,$$
$$[B_2]_п = 1:111=11:00111.$$

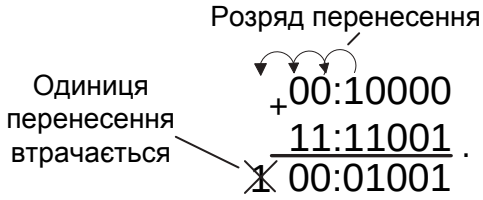
Відповідно до табл. 4.4 реалізується перетворення $A + (-B)$, тому другий операнд виразу перетворимо на додатковий код з урахуванням правила 3, тобто

$$[B_2]_п = 11:00111,\ [B_2]_з = 11:11000,\ [B_2]_д = 11:11001,$$

при цьому, пам'ятаючи правило формування кодів, запишемо

$$[A_2]_п = [A_2]_о = [A_2]_д = 00: 10000.$$

Виконаємо додавання за правилом 4 для додаткового коду:



У результаті отримано додатне число в додатковому коді: $[S_2]_д = [S_2]_п = 00:01001$. Його переведення в десяткову систему числення дає результат $S_2 = 00:01001 = +1001_2 = +9_{10}$.

Зазначимо, що при додаванні (відніманні) чисел може переповнюватися розрядна сітка суматора. Про це свідчить значення «01» у знаковому розряді при додатному переповненні розрядної сітки (у разі складання додатних чисел) або значення «10» – при від'ємному переповненні (при роботі з від'ємними числами).

Примітка *Під час переповнення розрядної сітки від операційного пристрою (див. рис. 2.1) до пристрою керування передається спеціальний сигнал-прапор – однорозрядне двійкове число, яке при переповненні набуває значення 1, а при відсутності переповнення – значення 0 [36].*

Дії з двійково-десятковими кодами чисел зводяться до операції алгебраїчного додавання окремих чисел, поданих двійковими тетрадами. Двійково-десяткові числа складаються в такій послідовності [4, 8]:

1. Додавання чисел починається з молодших цифр (тетрад) і виконується з урахуванням перенесень, що виникають, з молодших розрядів у старші.

2. Знак суми формується спеціальною логічною схемою за знаком більшого доданка.

3. Для того, щоб при додаванні двійково-десяткових кодів чисел виникали перенесення, аналогічні при додаванні чисел у десятковому поданні, проводять десяткову корекцію. Для цього до кожної тетради першого числа додається додатково по $6_{10} = 0110_2$.

4. Після операції підсумовування виконується корекція суми. З тих тетрад суми, в яких не було перенесень, вилучаються раніше внесені надлишки $6_{10} = 0110_2$. Для цього проводиться друга корекція. Операція віднімання замінюється операцією додавання з числом -6, поданим додатковим кодом – $6_{10} = 1010_2$, але тільки в тих розрядах, у яких не було перенесень. При цьому під час другої корекції можливі перенесення із тетрад блокуються.

Приклад. Додати два числа $A_{10} = 177$ і $B_{10} = 418$, подані в двійково-десятковому коді.

Подамо числа у вигляді двійково-десяткових чисел:

$$A_{2\text{-}10} = 0001\ 0111\ 0111 \text{ і } B_{2\text{-}10} = 0100\ 0001\ 1000.$$

Проведемо першу корекцію числа А (додамо до нього число 666):

$$\begin{array}{r} + \overset{\curvearrowleft\curvearrowleft\ \curvearrowleft\curvearrowleft}{0001\ 0111\ 0111} \\ \underline{0110\ 0110\ 0110} \\ A^*= 0111\ 1101\ 1101 \end{array}.$$

Додамо два числа $A^* + B_{2\text{-}10}$:

$$\begin{array}{r} + \overset{\curvearrowleft\ \ \curvearrowleft\curvearrowleft}{0111\ 1101\ 1101} \\ \underline{0100\ 0001\ 1000} \\ C^*= 1011\ 1111\ 0101 \end{array}.$$

Проводимо другу корекцію результату $C^* - [6_2]_д$ лише для тих розрядів, де не було перенесень з тетради в тетраду.

На попередньому етапі не було перенесення з тетрад, що

відповідають за сотні та десятки числа (тут отримано заборонені комбінації двійково-десяткової системи числення). Тому звідси вилучимо раніше внесені надлишки: проведемо додавання з числом -6, поданим додатковим кодом, тобто з числом $-6=[0110_2]_{\text{д}}=1010$:

Перенесення із тетрад заблоковано

$$\begin{array}{r} 1011\ 1111\ 0101 \\ +\ 1010\ 1010 \\ \hline C_{2\text{-}10}=\ 0101\ 1001\ 0101 \end{array}.$$

Звідси, C_{10} = 595.

Операція віднімання реалізується досить своєрідно. За загальним правилом додавання (пп. 1 – 4) до тетрад числа з великим модулем додаються додаткові коди тетрад іншого числа. При цьому перша корекція не проводиться, тому що в доповненнях тетрад вона враховується автоматично. Знак результату визначається за знаком числа з великим модулем [8].

Використання формату чисел з рухомою точкою істотно ускладнює схему арифметико-логічного пристрою. З огляду на те, що дія стандарту IEEE 754 поширюється на більшість програмних засобів і систем, то й алгоритми арифметичних дій з цими числами ускладнюються [4, 37]. Так, необхідно:

– порядки і мантиси чисел, що беруть участь в операціях, обробляти окремо;

– операцію додавання (віднімання) проводити з перетворенням чисел на додатковий код (зворотний код використовувати не можна, тому що «0» у такому поданні має двояке значення).

Згідно зі структурою машинної команди до початку виконання арифметичної дії операнди операції розміщуються у відповідні регістри арифметико-логічного пристрою.

Для наочної ілюстрації алгоритмів арифметичних дій з числами в нормальній формі будемо використовувати [4]:

– числа, подані в ненормалізованому експоненціальному вигляді, тобто числа, мантиса яких знаходиться в діапазоні $0{,}1<|m_a|<1$;

– істинний порядок числа p_a;

– числа, не зведені до рекомендованих форматів стандарту IEEE 754.

Таким чином, число з рухомою точкою запишемо у вигляді пари чисел:

$$A_2=\{\pm p_a;\pm m_a\}, \qquad (4.4)$$

тобто у вигляді послідовності цифр, що складається з двох підпослідовностей.

До першої входить частина цифр мантиси, що знаходяться після

точки; до другої – цифри істинного порядку. Для коду знаків, як і раніше, будемо використовувати подвоєне значення нуля при кодуванні знака «+» і подвоєне значення одиниці при кодуванні знака «-».

Алгоритм додавання (віднімання) чисел з рухомою точкою такий [4, 8]:

1. Вирівнюються порядки:

а) порівнюються порядки вихідних чисел шляхом їх віднімання: $\Delta p = p_1 - p_2$. При цьому важливо пам'ятати, що операція віднімання замінюється операцією додавання відповідно до табл. 4.1. Дії з доданками виконуються в додатковому коді за загальними правилами;

б) якщо різниця порядків дорівнює нулю, то однойменні розряди мантиси мають однаковий двійковий порядок, а отже, операція вирівнювання не проводиться;

в) якщо різниця порядків Δp є відмінною від нуля, то мантиса числа з меншим порядком зсувається в своєму регістрі вправо на кількість розрядів, що дорівнює отриманій різниці порядків Δp; після кожного зсуву порядок збільшується на одиницю.

2. Після вирівнювання порядків мантиси чисел можна складати (віднімати) залежно від необхідної операції. Дії з доданками виконуються за загальними правилами.

3. Порядок результату вибирається таким, що дорівнює більшому порядку.

Якщо мантиса результату є ненормалізованою, то здійснюються нормалізація і корекція порядку.

Приклад. Додати два числа $A_{10} = 1{,}375$ і $B_{10} = -0{,}625$.

Подамо числа в двійковій системі числення в нормальній формі відповідно до виразу (3.5):

$$A_2 = +1.011_2 = +0.1011 \cdot \exp_2 1 \text{ та } B_2 = -0.101_2 = -0.101 \cdot \exp_2 0.$$

З урахуванням формули (4.4) в прямому коді вихідні числа мають такий вигляд:

	Порядок	Мантиса
$[A_2]_п$	00:1	00:1011
$[B_2]_п$	00:0	11:101.

1. Порівнюємо порядки вихідних чисел шляхом знаходження різниці Δp.

У комп'ютері операція віднімання реалізується через додавання з переведенням чисел в додатковий код, тобто $\Delta p = p_a - p_b = p_a + (-p_b)$. Ураховуючи, що нуль в додатковому коді має єдине значення, запишемо

$$\begin{array}{r} [p_a]_д = 00{:}1 \\ +\ [p_b]_д = 00{:}0 \\ \hline \Delta p = 00{:}1 \end{array}.$$

Отже, $\Delta p \neq 0$, тому слід проводити вирівнювання порядків.

Різниця Δp є додатним числом, отже, порядок числа А більший за порядок числа В, тому мантису числа В зсунемо вправо на 1 розряд. Число В після зсуву матиме такий вигляд:

$$[B_2]_п = \{00:1;\ 11:0101\}.$$

2. Додаємо мантиси.

При переведенні чисел у додатковий код слід пам'ятати, що вигляд додатного числа в прямому коді збігається з його додатковим кодом, тому мантиса числа А буде такою:

$$[m_a]_п = 00:1011 = [m_a]_д = 00:1011.$$

Число В – від'ємне, тому в додатковому коді його мантиса матиме вигляд

$$[m_b]_п = 11:0101\ [m_b]_д = 11:1011.$$

При додаванні додаткових кодів чисел отримаємо такий результат:

$$[S_2]_д^m = \begin{array}{r} 00:1011 \\ +\ 11:1011 \\ \hline 00:0110 \end{array}.$$

Мантиса результату додавання є додатною, отже, її вигляд в додатковому коді збігається з прямим і не потребує додаткових перетворювань.

4. Порядок результату S дорівнює порядку числа з найбільшим порядком, тобто $p_S = 1$. У прямому коді сума двох чисел матиме такий вигляд:

Порядок	Мантиса
$[S_2]_п = 00:1$	00: 0110.

Мантиса є ненормалізованою, тому що значення її старшого розряду дорівнює 0.

5. Нормалізуємо результат додавання шляхом зсуву мантиси на 1 розряд вліво і відповідно віднімемо від значення порядку одиницю, тобто $[S_2]_п = \{00:0;\ 00:110\}$. Звідси $S_2 = 0{,}110$, отже, $S_{10} = 0{,}75$.

Операція множення найбільш просто реалізується в прямому коді (табл. 4.3) за правилами, аналогічними правилам множення десяткових чисел [18].

Таблиця 4.3

Двійкова таблиця множення однорозрядних чисел

A	B	Підсумок
0	0	0
0	1	0
1	0	0
1	1	1

Незалежно від форми подання чисел спочатку вміст старших розрядів (знаки чисел) складається на одноразрядному суматорі. При такому складанні перенесення, якщо воно виникає, втрачається, і добутку приписується знак згідно з відомим правилом алгебри.

Потім числа перемножуються. При цьому слід пам'ятати [4, 38], що при множенні n-розрядних множників розрядність добутку збільшується:

$$n + n = 2n. \qquad (4.5)$$

Приклад. Помножити два числа $A_{10} = 9$ та $B_{10} = 8$.

Запишемо вихідні числа в прямому коді:

$$[A_2]_п = 00{:}1001 \text{ та } [B_2]_п = 00{:}1000.$$

Перемножимо їх:

$$\begin{array}{r} \times\ 1001 \\ \underline{\quad 1000} \\ 00000000 \text{ (зсув на 0 розрядів)} \\ 00000000 \text{ (зсув на 1 розряд)} \\ 00000000 \text{ (зсув на 2 розряди)} \\ \underline{01001000 \text{ (зсув на 3 розряди)}} \\ S_2 = \underbrace{0\mathbf{1001000}}_{\text{розрядність добутку відповідно до формули (4.5)}} \end{array}$$

Знак добутку формуємо додаванням знакових розрядів, він дорівнює «00». Відповідь: $S_2 = 00{:}1001000 = +1001000$ або $S_{10} = +72$.

Як показує аналіз часу виконання окремих операцій, часові витрати на множення можуть досягати 80 % усього машинного часу при розв’язанні певних класів задач. Тому розроблення алгоритмів прискореного множення, які забезпечують час виконання операції, менший за тривалість у n кроків додавання і зсуву, має дуже важливе значення. Існує велика кількість таких алгоритмів, але їх основу становлять базові способи – логічні, апаратні та комбіновані.

Логічними називають такі способи, де ефект прискорення досягається шляхом ускладнення засобів керування процесом множення без будь-яких істотних змін у структурі арифметичних засобів.

Апаратними називають такі способи, які забезпечують прискорення множення через введення додаткового обладнання для поєднання в часі окремих складових процесу множення.

Комбіновані способи основані на поєднанні наведених вище підходів. Найпростішим серед них є пропуск кроків додавання у випадках, коли чергова цифра множника дорівнює нулю, тобто частинні добутки, отримані множенням на нуль, ігноруються.

Зазначимо, що завдяки своїй простоті та очевидності ця ідея дає змогу в середньому вдвічі зменшити кількість додавань, які припадають на один розряд множника (за умови, що 0 і 1 зустрічаються в будь-якому

розряді множника з однаковою частотою) [38].

Операція множення чисел з рухомою точкою також потребує різних дій з порядком і мантисою. Алгоритм цієї операції такий:

1. При множенні порядки додаються так само, як це робиться з числами з фіксованою точкою.

2. Мантиси множаться в прямому коді. Слід пам'ятати, що при множенні n-розрядних множників розрядність добутку повинна дорівнювати $2n$.

Приклад. Помножити два числа $A_{10} = 3{,}75$ та $B_{10} = -5$.

Подамо числа в двійковій системі числення в нормальній формі відповідно до виразу (3.5):

$$A_2 = +11.11_2 = +0.1111 \cdot \exp_2 10 \text{ та } B_2 = -101_2 = -0.101 \cdot \exp_2 11.$$

З урахуванням формули (4.4) в прямому коді вихідні числа мають такий вигляд:

	Порядок	Мантиса
$[A_2]_п$	00:10	00:1111
$[B_2]_п$	00:11	11:1010.

1. Додамо порядки:

$$\begin{array}{r} 00{:}11 \\ +\ 00{:}10 \\ \hline p_s = 01{:}01 \end{array}$$

Ознака переповнення розрядної сітки

Збільшення розрядності сітки

$$\begin{array}{r} 00{:}011 \\ +\ 00{:}010 \\ \hline p_s = 00{:}101 \end{array}.$$

2. Для виконання умови (4.5) допишемо справа незначущі нулі до мантиси числа B:

$$[m_a]_п = 1111 \text{ та } [m_b]_п = 1010.$$

Помножимо мантиси (для прикладу множення на 0 не ігноруємо):

$$\begin{array}{r} 1111 \\ \times\ 1010 \\ \hline 0000 \\ 1111 \\ 0000 \\ 1111 \\ \hline S_2 = 10010110 \end{array}$$

(зсув на 0 розрядів)
(зсув на 1 розряд)
(зсув на 2 розряди)
(зсув на 3 розряди)

З урахуванням умови (4.5) зазначимо, що при множенні двох чотирирозрядних чисел отримали восьмирозряде число, тобто результат є коректним.

3. Сформуємо знак добутку додаванням відповідних знакових розрядів: 00+11 = 11.

У результаті отримаємо число $[S_2]_п$ = {00:101; 11:10010110}, переведення якого в форму з фіксованою точкою дає двійкове число S_2 = -10010,11, тобто S_{10} = -18,75.

Було розглянуто алгоритми виконання арифметичних дій, орієнтовані на машинні коди. Зазначимо, що зараз програмування в машинному коді здійснюють дуже рідко через громіздкість і трудомісткість ручного керування ресурсами процесора, за винятком ситуацій, коли потрібна екстремальна оптимізація. Тому переважна більшість програм пишеться мовами вищого рівня та транслюється в машинний код компіляторами [36].

3. Логічні основи комп’ютерних технологій

Робота сучасних комп’ютерів зводиться до оброблення і пересилання послідовності нулів і одиниць, якими закодовано різну інформацію. Оскільки основна система числення будь-якої інформаційної технології – двійкова, то зручним математичним апаратом для опису функціонування апаратних засобів комп'ютера є алгебра логіки [39].

Алгебра логіки – розділ математичної логіки, який вивчає висловлювання та логічні операції над ними. Алгебра логіки допомагає зрозуміти внутрішню будову комп'ютера. За її законами проєктують різні частини комп'ютерів, зокрема пам'ять та процесор, а на етапі конструювання апаратних засобів вона дає змогу значно спрощувати логічні функції, що описують функціонування схем комп'ютера, і зменшувати кількість елементарних логічних елементів, з десятків тисяч яких складаються основні вузли комп'ютера [40].

Фізичну схему вузлів комп’ютера реалізують з використанням логічних елементів. Логічні елементи комп’ютера – це частини електронної схеми, що реалізують логічні операції «І», «АБО», «НІ», набір яких наведено в табл. 4.4. Кожна така частина має один або кілька входів і виходів, через які проходять електричні сигнали – електричні імпульси. Ці сигнали прийнято позначати цифрами «0» і «1»: є імпульс – логічне значення 1, немає імпульсу – значення 0. Кожен елемент має своє умовне позначення, яке характеризує його логічну функцію, але не вказує на те, яка саме електронна схема в ньому реалізована, що спрощує запис і розуміння складних логічних схем [39, 40].

Таким чином, на входи логічних елементів надходять сигнали – операнди, на виході з'являється сигнал – результат операції. Перетворення сигналу логічним елементом описують за допомогою таблиць істинності, яка є еквівалентною таблиці істинності, що відповідає логічній функції [4].

Таблиця істинності – це табличне подання логічної схеми (операції), в якому перелічено всі можливі поєднання значень істинності вхідних сигналів (операндів) разом зі значеннями істинності вихідного сигналу (результату операції) для кожного з цих поєднань. Таблицю істинності складають за такими правилами [39]:

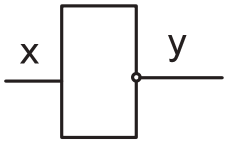
– кількість рядків визначають як 2^n (де n – кількість вхідних операндів) плюс додатково два рядки для головки таблиці;

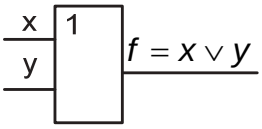
– кількість стовпців характеризується кількістю операндів і числом логічних операцій схеми.

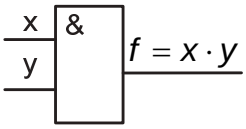
При побудові таблиці необхідно врахувати всі можливі варіанти об'єднання вхідних логічних 0 та 1. Потім слід визначити порядок виконання логічних операцій у схемі та побудувати таблицю з урахуванням таблиць істинності елементарних логічних операцій, наведених у табл. 4.4.

Таблиця 4.4

Схеми логічних елементів, що виконують елементарні логічні операції

<table>
<tr><th>Схема</th><th>Таблиця істинності логічного елемента</th></tr>
<tr><td> 

Інверсія – логічне заперечення (операція «НІ»)</td><td>
<table>
<tr><td>x</td><td>1</td><td>0</td></tr>
<tr><td>$f = \overline{x}$</td><td>0</td><td>1</td></tr>
</table>
</td></tr>
<tr><td> 

Диз’юнкція – логічне додавання (операція «АБО»)</td><td>
<table>
<tr><td>x</td><td>y</td><td>$f = x \vee y$</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
<tr><td> 

Кон’юнкція – логічне множення (операція «І»)</td><td>
<table>
<tr><td>x</td><td>y</td><td>$f = x \cdot y$</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
</table>

За допомогою подібних умовних позначень можна реалізувати будь-яку логічну функцію, що описує роботу комп'ютера, яку в подальшому формалізують (замінюють логічною формулою).

У найпростішому випадку логічною формулою можна вважати будь-яку змінну, символи «істинна» або «1» та «хибно» або «0». Якщо змінні A і B – логічні формули, то $\overline{A}$, $A \vee B$, $A \cdot B$ також логічні формули [39].

Приклад. Контролер мережі Modbus опитує три slave-пристрої та формує сигнал аварії у випадку, якщо значення сигналів з двох slave-пристроїв є нижчими за встановлену межу. Записати ситуацію аварії у

вигляді логічної формули.

Сформуємо такі логічні змінні:

A – «Сигнал зі slave-пристрою 1 нижче за встановлену межу»;

B – «Сигнал зі slave-пристрою 2 нижче за встановлену межу»;

C – «Сигнал зі slave-пристрою 3 нижче за встановлену межу».

Аварійна ситуація: X – «Сигнал з двох slave-пристроїв нижче за встановлену межу», тобто X – «Сигнали зі slave-пристрою 1 і slave-пристрою 2 нижче за встановлену межу», або «Сигнали зі slave-пристрою 2 і slave-пристрою 3 нижче за встановлену межу», або «Сигнали зі slave-пристрою 1 і slave-пристрою 3 нижче за встановлену межу».

Логічна формула має вигляд

$$X = A \cdot B \vee B \cdot C \vee A \cdot C. \qquad (4.6)$$

За логічною формулою будують схеми. При цьому порядок реалізації пристроями комп'ютера логічних операцій задається у формулі круглими дужками, але для зменшення кількості дужок прийнято вважати, що спочатку виконується інверсія «НІ», потім – кон'юнкція «І», а після – диз'юнкція «АБО».

Приклад. Відомо, що виконання будь-яких операцій у комп’ютері зводиться до додавання та зсуву, що робить суматор головним елементом процесора. Побудувати логічну схему суматора (без урахування розряду перенесення з молодшого розряду), правила роботи якого задаються відповідно до таблиці додавання (табл. 4.5) [1].

Таблиця 4.5

Правила додавання двох однорозрядних двійкових чисел

Доданки		Перенесення в наступний розряд C	Сума S
A	B		
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Запишемо логічну формулу, що реалізує операцію додавання відповідно до правил табл. 4.5.

Зазначимо, що вихід C можна реалізувати з використанням операції логічного множення, тобто

$$C=A \cdot B.$$

Аналізуючи останній стовпець табл. 4.5, зазначимо, що він майже повністю збігається з результатами виконання операції логічного додавання (див. табл. 4.4), крім випадку, коли на вхід подаються дві логічні одиниці: за табл. 4.5 на виході в цьому випадку слід отримати логічний 0. Потрібний результат можна отримати, якщо результат логічного додавання помножити на інверсне значення виходу C. Таким чином, отримати вихід S можна із застосуванням такої логічної функції:

$$S = (A \vee B) \cdot \overline{(A \cdot B)}. \qquad (4.7)$$

Для перевірки правильності отриманої логічної формули побудуємо для неї таблицю істинності (табл. 4.6).

Таблиця 4.6

Таблиця істинності для формули (4.7)

Вхідні змінні		Проміжні формули			Вихідний результат S
A	B	A∨B	A·B	$\overline{A \cdot B}$	$S = (A \vee B) \cdot \overline{(A \cdot B)}$
0	0	0	0	1	0
0	1	1	0	1	1
1	0	1	0	1	1
1	1	1	1	0	0

Порівнюючи вихідні результати табл. 4.5 та 4.6, зазначимо, що вони є ідентичними. Тому, ґрунтуючись на одержаній логічній формулі (4.7), побудуємо із базових логічних елементів схему додавання однорозрядних двійкових чисел (рис. 4.2).

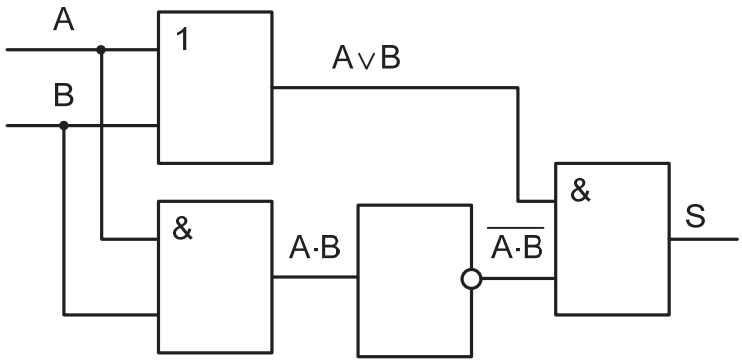


Рис. 4.2. Логічна схема додавання двійкових чисел відповідно до правил табл. 4.5

Схему, наведену на рис. 4.2, називають напівсуматором, оскільки вона реалізує додавання однорозрядних двійкових чисел A і B без урахування перенесення з молодшого розряду. Для додавання однорозрядних двійкових чисел з урахуванням перенесення (див. табл. 4.1) використовується суматор, який складається з напівсуматорів.

Для складних логічних схем буває важко одразу визначити доцільність їх створення. Зазначимо, що логічна схема має сенс лише в тому випадку, якщо її логічна формула є здійсненною.

Здійсненною називають таку логічну формулу, яка набуває значення або логічного нуля, або логічної одиниці залежно від різних можливих варіантів поєднання логічних 0 та 1 вхідних змінних. Для того, щоб визначити здійсненність логічної формули, а отже, доцільність створення логічного пристрою, складають таблицю істинності [12].

Приклад. Визначити, чи є здійсненною логічна формула (4.6), отримана для контролера мережі Modbus.

Складемо таблицю істинності для вихідної формули (табл. 4.7).

Таблиця 4.7

Таблиця істинності для формули (4.6)

Вхідні змінні			Проміжні формули				Вихідний результат X
A	B	C	A·B	B·C	A·B∨B·C	A·C	A·B∨B·C∨ A·C
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	1	0	1	1	0	1
1	0	0	0	0	0	0	0
1	0	1	0	0	0	1	1
1	1	0	1	0	1	0	1
1	1	1	1	1	1	1	1

Зазначимо, що для всіх можливих варіантів значень вхідних змінних логічна формула може набувати значення або логічного нуля, або логічної одиниці. Тому формула є здійсненною.

Побудуємо за логічною формулою схему (рис. 4.3) з використанням найпростіших логічних елементів табл. 4.4.

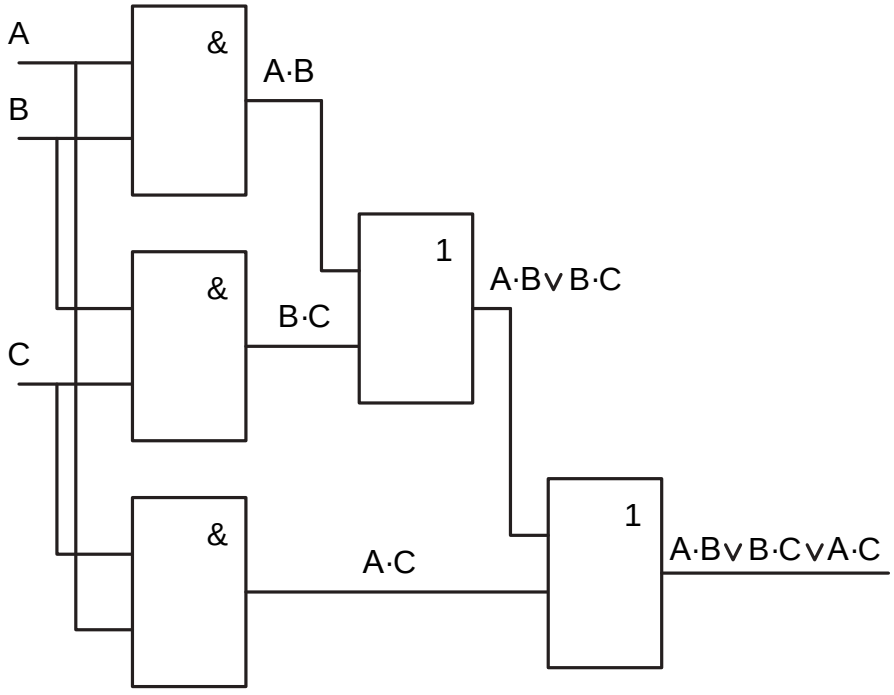


Рис. 4.3. Схема логічного пристрою

Під спрощенням, або мінімізацією, формули розуміють рівносильне перетворення, унаслідок якого можна отримати формулу, що порівняно з вихідною або містить меншу кількість операцій кон'юнкції і диз'юнкції і не має заперечень неелементарних формул, або містить меншу кількість вкладених змінних.

Існує кілька законів алгебри логіки, що дають змогу мінімізувати вихідні функції (табл. 4.8) [39].

Приклад. Побудувати схему за виразом $F = x \cdot y \vee \overline{x} \cdot y \vee \overline{x} \cdot \overline{y}$. З використанням законів алгебри логіки мінімізувати F, побудувати схему

для отриманого виразу.

Таблиця 4.8

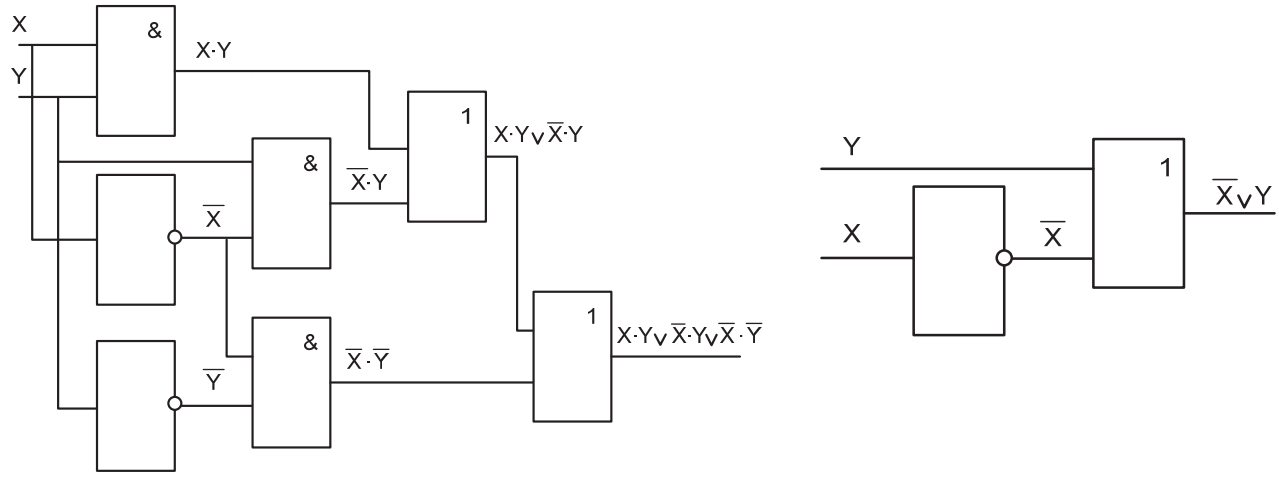
Закони алгебри логіки, що спрощують логічні формули

Закон	Для операцій	
	«АБО»	«І»
Переставний	$x \vee y = y \vee x$	$x \cdot y = y \cdot x$
Сполучний	$x \vee (y \vee z) = (x \vee y) \vee z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
Розподільний	$x \cdot (y \vee z) = x \cdot y \vee x \cdot z$	$x \vee (y \cdot z) = x \vee y \cdot x \vee z$
Правила де Моргана	$\overline{x \vee y} = \overline{y} \cdot \overline{x}$	$\overline{x \cdot y} = \overline{y} \vee \overline{x}$
Ідемпотентності	$x \vee x = x$	$x \cdot x = x$
Поглинання	$x \vee (x \cdot y) = x$	$x \cdot (x \vee y) = x$
Склеювання	$(x \cdot y) \vee (\overline{x} \cdot y) = y$	$(x \vee y) \cdot (\overline{x} \vee y) = y$
Згортки	$x \vee \overline{x}y = x \vee y$	$x(\overline{x} \vee y) = xy$
	$\overline{x} \vee xy = \overline{x} \vee y$	$\overline{x}(x \vee y) = \overline{x}y$
Операція змінної з її інверсією	$x \vee \overline{x} = 1$	$x \cdot \overline{x} = 0$
Операція з константами	$x \vee 0 = x$ і $x \vee 1 = 1$	$x \cdot 1 = x$ і $x \cdot 0 = 0$
Операція подвійного заперечення	$\overline{\overline{x}} = x$	

Схемау за вихідним виразом наведено на рис. 4.4, а. За даними табл. 4.8 спростимо його:

$$F = x \cdot y \vee \overline{x} \cdot y \vee \overline{x} \cdot \overline{y} = x \cdot y \vee \overline{x} \cdot (y \vee \overline{y}) = x \cdot y \vee \overline{x} = \overline{x} \vee y .$$

Схему за спрощеним виразом показано на рис. 4.4, б.



а б

Рис. 4.4. Схеми логічного пристрою: а – за вихідною формулою; б – за спрощеною формулою

Тестові запитання та завдання для самоперевірки

1. У комп’ютері машинні коду використовують:

а) для виконання арифметичних і логічних операцій;

б) з метою спрощення арифметичних операцій;

в) для здвигу інформаційних бітів праворуч або ліворуч на необхідну кількість розрядів.

2. Яке твердження є правильним:

а) додатковий код є доповненням прямого коду будь-якого числа до машинної одиниці;

б) зворотний код додатного числа містить одиницю в знаковому розряді, а значущі розряди числа замінюються інверсними;

в) результат арифметичної операції наводиться в тому коді, в якому подано вихідні складові;

г) машинний код дає змогу розділити правила для оброблення знакових і значущих розрядів чисел?

3. Зворотний код додатного числа дорівнює:

а) прямому коду цього числа;

б) інверсному значенню цього числа;

в) інверсному значенню цього числа, до молодшого розряду якого додано одиницю.

4. Додатковий код необхідний:

а) для вирівнювання розрядної сітки доданків;

б) для окремого оброблення порядків і мантис чисел, що беруть участь в операції додавання;

в) для виключення можливого випадку подвійного подання нуля;

г) для спрощення операції множення чисел.

5. Для чого здійснюють двійкову корекцію при додаванні двійково-десяткових чисел:

а) для спрощення алгоритму додавання двійково-десяткових чисел;

б) для здійснення необхідних перетворень кодів (з прямого на зворотний і додатковий);

в) для формування знака суми;

г) для того, щоб при додаванні виникали перенесення?

6. Множення двійкових чисел здійснюють за правилами:

а) 0*0 = 0; 0*1 = 1; 1*1 = 0; 1*0 =0;

б) 0*0 = 1; 0*1 = 0; 1*1 = 1; 1*0 =0;

в) 0*0 = 0; 0*1 = 0; 1*1 = 1; 1*0 =0.

7. Укажіть правильну послідовність алгоритму побудови таблиці істинності:

1) з'ясувати кількість стовпців;

2) визначити всі можливі варіанти об'єднання вхідних змінних;

3) з'ясувати кількість рядків у таблиці;

4) встановити порядок виконання логічних операцій;

5) заповнити таблицю істинності по стовпцях.

8. Продовжіть речення: «Частина електронної логічної схеми, яка реалізує елементарну логічну операцію, – це …:

а) логічна схема комп’ютера;

б) логічний елемент комп'ютера;

в) електронний компонент комп'ютера;

г) напівсуматор.

9. Виберіть правильне визначення:

а) здійсненна логічна формула – формула, яка набуває значення або логічного нуля, або логічної одиниці залежно від різних можливих варіантів поєднання логічних 0 та 1 вхідних змінних;

б) здійсненна логічна формула – формула, яка набуває значення лише логічної одиниці залежно від різних можливих варіантів поєднання логічних 0 та 1 вхідних змінних;

в) здійсненна логічна формула – формула, яка набуває значення лише логічного нуля залежно від різних можливих варіантів поєднання логічних 0 та 1 вхідних змінних.

10. Установіть відповідність «умова – формула» (наприклад, а – а, б – б та ін.):

Умова	Формула
а) змінні A і B складають логічну формулу, яка є хибною тоді і тільки тоді, коли обидві змінні є істинними	а) $f = \overline{A} \cdot \overline{B}$
б) змінні A і B складають логічну формулу, яка є істинною тоді і тільки тоді, коли хибною є або змінна A, або змінна B	б) $f = \overline{A \cdot B}$
в) змінні A і B складають логічну формулу, яка є хибною тоді і тільки тоді, коли істинною є або змінна A, або змінна B	в) $f = \overline{A} \vee \overline{B}$
г) змінні A і B складають логічну формулу, яка є істинною тоді і тільки тоді, коли обидві змінні є хибними	г) $f = \overline{A \vee B}$

11. Що є результатом виконання операції $x \vee x \cdot y$:

а) 1;

б) 0;

в) x;

г) y?

12. Наведіть машинний код чисел ± 16, ± 13.

13. З використанням машинних кодів додати числа A = 5 і B = 3.

14. Дослідіть алгоритм додавання двійково-десяткових чисел на прикладі додавання чисел A = 123 і B = 304.

15. Дослідіть алгоритм додавання чисел з рухомою точкою на прикладі додавання чисел A = 1,5 і B = -3,25.

16. З використанням машинних кодів помножте числа A = -5 і B = 3.

17. Дослідіть алгоритм множення чисел з рухомою точкою на прикладі множення чисел A = 2,5 і B = -0,5.

18. Пристрій задано логічною формулою $f = \overline{A} \cdot B \vee \overline{B} \cdot A \cdot \overline{C}$. Складіть схему.

19. Складіть таблицю істинності логічних формул $f = \overline{x} \cdot y \vee \overline{x \vee y} \vee x$ та $f = \overline{x \vee y} \cdot (x \cdot \overline{y})$.

20. Спростіть вихідні функції $f = \overline{\overline{a} \cdot a} \vee b \cdot (a \cdot b \vee b)$ і $f = a \cdot b \cdot (\overline{a} \vee \overline{b})$.

.

Додаток

ІНТЕРФЕЙС СИСТЕМИ КОМП’ЮТЕРНОЇ МАТЕМАТИКИ SCILAB

Scilab – система комп'ютерної математики, призначена для здійснення інженерних і наукових обчислень. Вільно розповсюджувану версію Scilab можна отримати на сайті www.scilab.org (рис. Д.1), де є версії для операційних систем Windows і Linux [41].

Рис. Д.1. Інтерфейс сайту www.scilab.org

Інтерфейс Scilab наведено на рис. Д.2, що складається з таких основних елементів: головне меню, оглядач файлів і змінних і командна консоль [32, 41].

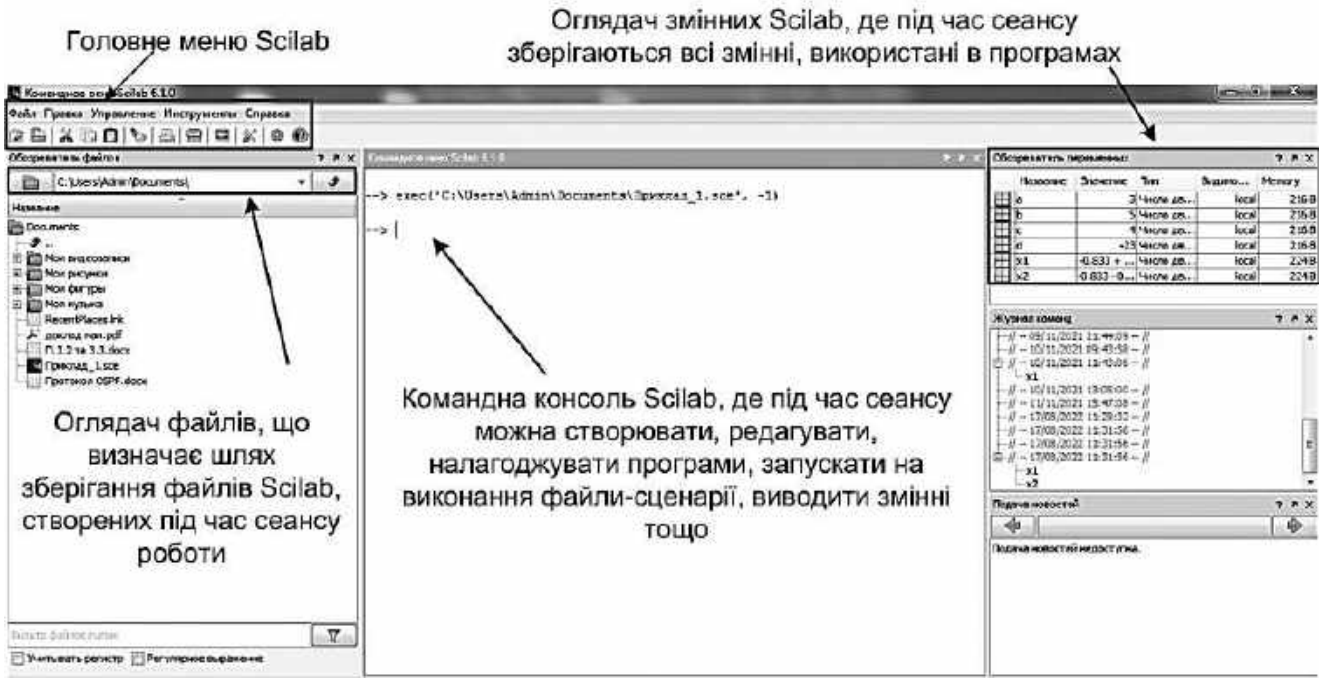


Рис. Д.2. Інтерфейс Scilab

Головне меню Scilab містить команди, призначені для роботи з файлами (зокрема меню SciNotes – для створення, редагування та налагодження файлів-сценаріїв), налаштування середовища, редагування команд поточної сесії й отримання довідкової інформації.

Файл-сценарій – це список команд Scilab, збережений у пам’яті

комп'ютера як файл з розширенням .sce. Його створюють у редакторі SciNotes з головного меню системи.

Вікно редактора файлів-сценаріїв має стандартний вигляд (рис. Д.3), складається із заголовка, меню, панелі інструментів, рядка стану. Редактор SciNotes дає можливість працювати з кількома вікнами (пункт меню Windows), має прийняті для текстових редакторів прийоми редагування (пункт меню Edit) і пошуку (пункт меню Search). Крім того, можна виконати налаштування середовища редактора (пункт меню Options), викликати довідкову інформацію (пункт меню Help) і здійснити налагодження програми, створеної в редакторі (пункт меню Debug). Вийти з режиму редагування можна просто, закривши вікно SciNotes або виконавши команду File – Exit [32, 41].

Рис. Д.3. Інтерфейс редактора SciNotes

За згодою файл-сценарій має ім'я Untitled1.sce. Для його зміни та збереження введеної інформації виконують команду «File - Save file As». Усі зміни, внесені в файл-сценарій, слід зберігати. Тільки після цього запускають обов'язкову команду виконання файлу [32].

Введення тексту у вікні редактора SciNotes здійснюють за правилами, прийнятими для команд Scilab.

На рис. Д.3 наведено приклад уведених команд програми для розв'язання квадратного рівняння $2x^2 + 6x + 3 = 0$.

Перші три рядки програми містять операції присвоєння та задають значення коефіцієнтів рівняння – змінні a, b, c. Для визначення змінної набирають її ім'я, символ «=» і значення змінної. Отже, у Scilab знак рівності – це оператор присвоєння, дія якого аналогічна подібним операторам будь-яких мов програмування. Система Scilab «чутлива до регістра», тобто розрізняє великі та малі літери в іменах змінних, а отже, ABC, abc, Abc або aBC – це імена різних змінних. Кожен з цих рядків закінчують символом «;», який ставлять після команд, що потребують

виведення значень [32, 42].

Вираз у правій частині оператора присвоєння може бути числом (як в перших рядках програми), арифметичним виразом, рядком символів або символьним виразом (у випадку використання символьної змінної вираз у правій частині оператора беруть в одинарні або подвійні лапки). За згодою, якщо немає додаткових указівок, всі числові змінні – дійсні з подвійною точністю, тобто мають тип double (довгий дійсний формат). Ціла частина числа відділяється від дробової символом «.». Як приклад, у четвертому рядку наведено операцію присвоєння з арифметичним виразом для розрахунку дискримінанту, а в рядках п'ять і шість – з виразами для знаходження коренів рівняння. У цих рядках наприкінці виразів теж указано символ «;». У цьому випадку результат обчислень не виводиться, а активізується наступний командний рядок.

Масиви даних (рядки символів) зазвичай подають у вигляді векторів або матриць [42].

Вектор Scilab – упорядкована сукупність елементів (одномірний масив) одного типу даних, коли до кожного елемента вектора можна звернутися за його унікальним порядковим номером (індексом). У системі Scilab усі індекси починаються з одиниці.

Під час створення вектора масив чисел подають у квадратних дужках, числа вказують через пробіл, наприклад:

t=[1971 1974 1978],

де t(2)=1974, тобто елемент вектора t з порядковим номером 2 дорівнює 1974.

Вектор, елементи якого є числовою послідовністю типу арифметичної прогресії, може бути створений з використанням конструкції вигляду

<початкове значення>:<крок>:<кінцеве значення>

за умови, що крок має неодиничне значення. Якщо крок дорівнює 1, його не вказують.

Матриця Scilab – це двовимірний масив однотипних елементів або декілька векторів-рядків.

У загальному випадку синтаксична конструкція матриці має вигляд

$A=[x_{11}\ x_{12}\ \ldots\ x_{1n};\ x_{21}\ x_{22}\ \ldots\ x_{2n};\ \ldots;\ x_{n1}\ x_{n2}\ \ldots\ x_{nn}]$,

тобто в квадратних дужках поєднують вектори-рядки, які відокремлюють один від одного крапкою з комою.

У Scilab передбачено функцію «Текстовий коментар» – це частина рядка, що починається з символів «//». Частина рядка після «//» не сприймається як команда, натискання клавіші Enter приводить до активізації наступного командного рядка [32, 41].

Для зменшення кількості помилок під час створення програм у файлі-сценарії стандартні функції (табл. Д.1) виділяють кольором. Наприклад, на рис. Д.3 функцію «Корінь ($\sqrt{}$)» (у Scilab вбудована функція «sqrt») у

файлі-сценарії виділено блакитним кольором, а функцію «Текстовий коментар» (у Scilab функція «//») – зеленим разом із текстом.

Таблиця Д.1

Поширені стандартні функції в Scilab

Функція	Оператор	Значення	Функція	Оператор	Значення
X=Y	X=Y	X присвоюється значення Y	$\sqrt{X}$	sqrt (X)	Добування квадратного кореня з X
X+Y	X+Y	До X додається Y	$\lvert X \rvert$	abs(X)	Абсолютне значення X
X-Y	X-Y	Від X віднімається Y	e^x	exp(X)	Експонента числа X
XY	X*Y	X множиться на Y	ln(X)	log(X)	Натуральний логарифм X
$\frac{X}{Y}$	X/Y	X діліться на Y	lg(X)	log10(X)	Десятковий логарифм X
X^n	X^n	X підноситься до степеня n	$\log_2(X)$	log2(X)	Двійковий логарифм X

У випадку наявних помилок у файлі-сценарії в командній консолі наводиться відповідне службове повідомлення із зазначенням рядка програми, де зафіксовано помилку (рис. Д.4).

```
--> exec('C:\Users\Admin\Documents\Приклад_1.sce', -1)
на строке     5 исполняемого файла C:\Users\Admin\Documents\Приклад_1.sce

Неопределённая переменная: sqrt

-->
```

Рис. Д.4. Службове повідомлення Scilab у випадку наявних помилок

Функції в Scilab визначають ім'ям і значенням аргументу (аргументів) у круглих дужках. При цьому [41]:

– ім'я функції (на відміну від ім’я змінної) не чутливе до регістра, наприклад, використання імен функції abs, ABS або Abs дає однаковий ефект;

– аргументи функції обов’язково поміщують у круглі дужки;

– можна застосовувати вкладення функцій одна в іншу.

Під час виконання програми файлу-сценарію послідовно заповнюються рядки оглядача змінних (див. рис. Д.2). Отже, результат виконання можна знайти в цьому меню або вивести в командну консоль з використанням функцій (рис. Д.5) [42]:

– **disp**('text', a), наприклад:

disp('Значення першого кореня x1=', x1),

де a – змінна, значення якої необхідно вивести на екран під час виконання функції; 'text' – текст-пояснення, що виводиться на екран під час виконання функції;

– **printf**('text a=%f\n', a), наприклад:

printf('Значення першого кореня x1=%f\n', x2),

де a – змінна, значення якої необхідно вивести на екран; 'text a=%f\n' – текст-пояснення; %f – позначення місця виведення окремої змінної на екран (якщо слід вивести рядок змінних, то символ «f» замінюють символом «s», тобто «%s»); \n – позначення необхідності подальшого перенесення тексту після функції на наступний рядок.

```
"Значення першого кореня x1 ="

-0.6339746

"Значення другого кореня x2 ="

-2.3660254
```

а

```
Значення першого кореня x1 =-0.633975
Значення першого кореня x2 =-2.366025
```

б

Рис. Д.5. Приклад виведення в командну консоль результатів обчислень: а – з використанням функції **disp**(); б – з використанням функції **printf**()

Більш наочним є графічне подання результатів розрахунків. Для побудови графіка функції y(x) необхідного вигляду та кольору застосовують функцію plot(x,y) або plot2d(x,y), зокрема

plot(x,y,'s'),

де s – необов'язковий аргумент, який використовують всередині функції plot для налаштування вигляду кожної нової лінії шляхом поєднання інформації про стиль і колір ліній або маркери (табл. Д.2).

Таблиця Д.2

Деякі елементи типу ліній функції plot у Scilab

Стиль лінії		Колір лінії		Тип маркера	
Позначка для s	Визначення	Позначка для s	Визначення	Позначка для s	Визначення
--	Штрихова лінія	r	Червоний	*	Зірочка
:	Пунктирна лінія	g	Зелений	°	Коло
-.	Штрихпунктирна лінія	b	Синій	^	Трикутник

Якщо в одному графічному вікні необхідно побудувати декілька функцій $y_i(x)$, використовують функцію plot(x,y_1,'s_1',x,y_2,'s_2', …, x,y_i,'s_i').

Для зручного подання графічної інформації ці функції використовують сумісно з набором додаткових команд [32, 41]:

- title('Текст') – встановлює заголовок до графіка;
- xlabel 'Текст' і ylabel 'Текст' – підписує назву осей x і y відповідно;
- xgrid(2) – наносить координатну сітку;
- legend('Текст',pos=4) – розміщує в графічному вікні легенду до рисунка, де pos означає позицію легенди (1 – справа вверху, 2 – зліва вверху; 3 – справа внизу, 4 – зліва внизу).

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Гуржій, А. М. Інформатика та інформаційні технології [Текст] : підручник / А. М. Гуржій, Н. І. Поворознюк, В. В. Самсонов. – Харків : Компанія СМІТ, 2007. – 352 с.

2. Комп'ютери та комп'ютерні технології [Текст] : навч. посіб. / Ю. Б. Бродський, К. В. Молодецька, О. Б. Борисюк, І. Ю. Гринчук. – Житомир : Житомир. нац. агроеколог. ун-т, 2016. – 186 с.

3. Геоінформаційні системи [Електронний ресурс]. – Режим доступу: http://distance.edu.vn.ua/books/Inform. – 06.10.2021.

4. Даншина, С. Ю. Комп'ютерні технології для ГІС-додатків [Текст]: навч.-метод. посіб. до практ. і лаб. занять / С. Ю. Даншина. – Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2020. – 56 с.

5. Страницы истории отечественных ИТ. Т. 1 / сост. Э. М. Пройдаков. – М. : Альпина Паблишер, 2015. – 265 с.

6. Орлов, С. «Мэйнфреймы навсегда». Почему проекты их замены часто заканчиваются неудачно [Электронный ресурс] / С. Орлов. – Режим доступа: https://www.cnews.ru/articles . – 06.11.2021.

7. Агольцов, А. Как мэйнфреймы не вымерли [Электронный ресурс] / А. Агольцов. – Режим доступа: https://habr.com/ru/post/554100/. – 05.05.2022.

8. Пятибратов, А. П. Вычислительные системы, сети и телекоммуникации [Текст]: учеб. для вузов / А. П. Пятибратов, Л. П. Гудынго, А. А. Кириченко. – М. : Финансы и статистика, 2003. – 512 с.

9. Altair 8800: короткий рассказ о великом компьютере [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/cloud_mts/blog/551814/. – 05.06.2021.

10. IBM PC: полная история [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/406013/. – 05.06.2021.

11. Олифер, В. Г. Компьютерные сети. Принципы, технологии, протоколы [Текст] / В. Г. Олифер, Н. А. Олифер. – СПб. : Питер, 2004. – 864 с.

12. Скрипин, В. Технологии будущего или когда «умрет» закон Мура? [Электронный ресурс] / В. Скрипин. – Режим доступа: https://hi-news.ru/technology/ – 15.06.2021.

13. Закон количества транзисторов в процессорах (закон Мура) в цифрах и графиках [Электронный ресурс]. – Режим доступа: https://pro-spo.ru/vse-o-kompyuterax-i-noutbukax/. – 15.06.2021.

14. Аноприенко, А. Я. Обобщения закона Мура [Текст] / А. Я. Аноприенко // Информатика и кибернетика. – 2017. – № 3(9). – С. 14 – 21.

15. Закон Мура против нанометров [Электронный ресурс]. – Режим доступа: https://www.ixbt.com/cpu/microelectronics.shtml. – 15.06.2022.

16. Как работают квантовые компьютеры. Собираем пазл [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/480480/. – 15.06.2022.

17. Сергієнко, А. М. Архітектура комп'ютерів [Текст] : консп. лекцій / А. М. Сергієнко. – Київ : НТУУ«КПІ», 2015. – 198 с.

18. Борян, Л. О. Комп'ютери та комп'ютерні технології [Текст] : консп. лекцій / Л. О. Борян. – Миколаїв : Миколаїв. нац. аграрний ун-т, 2019. – 138 с.

19. Машинное слово [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/. – 25.06.2021.

20. АЛУ на логических микросхемах [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/595367/. – 25.06.2021.

21. Криницкий, Н. А. Программирование и алгоритмические языки [Текст] / Н. А. Криницкий, Г. А. Миронов, Г. Д. Фролов. – М. : Наука, 1975. – 496 с.

22. Как общаются машины: протокол Modbus [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/advantech/blog/450234/ – 25.06.2022.

23. Гордеев, А. В. Операционные системы [Текст] : учеб. для вузов / А. В. Гордеев. – СПб. : Питер, 2007. – 416 с.

24. Андреева, Е. Информатика: системы счисления и компьютерная арифметика [Текст] : учеб. для вузов / Е. Андреева, И. Фалина. – М. : Лаборатория Базовых Знаний, 2002. – 336 с.

25. Представление целых чисел [Электронный ресурс]. – Режим доступа: https://help.scilab.org/docs/6.1.0/ru_RU/dec2bin.html – 28.07.2022.

26. Единицы измерения ёмкости носителей и объёма информации [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/ – 25.06.2022.

27. Scan-код [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/ – 25.06.2022.

28. Петцольд, Ч. Код. Тайный язык информатики [Текст] / Ч. Петцольд. – М. : Манн, Иванов и Фербер, 2019. – 448 с.

29. Двобайтовий набір символів [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/DBCS. – 25.06.2022.

30. Unicode [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/ – 25.06.2022.

31. UTF-16 [Электронный ресурс]. – Режим доступа: https:// https://ru.wikipedia.org/wiki/UTF-16 – 25.08.2022.

32. Алексеев, Е. Р. Scilab. Решение инженерных и математических задач [Текст] / Е. Р. Алексеев, О. В. Чеснокова, Е. А. Рудченко. – М. : БИНОМ. Лаборатория знаний, 2008. – 260 с.

33. FLOPS [Электронный ресурс]. – Режим доступа: https:// https://ru.wikipedia.org/wiki/FLOPS. – 25.01.2022.

34. Яшкардин, В. Л. IEEE 754: стандарт двоичной арифметики с плавающей точкой [Электронный ресурс] / В. Л. Яшкардин. – Режим доступа: http://www.softelectro.ru/ieee754.html. – 20.02.2022.

35. Butterfield, A. A Dictionary of Computer Science [Text] / A. Butterfield, G. E. Ngondi, A. Kerr. – Oxford University Press, 2016. – 638 p.

36. Юров, В. И. Assembler [Текст]: учеб. пособие / В. И. Юров – СПб. : Питер, 2003. – 637 с.

37. Лужецький, В. А. Арифметичні основи комп'ютерної техніки [Текст]: навч. посіб. / В. А. Лужецький, Ю. Г. Лега. – Черкаси : ЧДТУ, 2007. – 203 с.

38. Чегренець, В. М. Комп'ютер та комп'ютерна арифметика [Текст] : навч. посіб. / В. М. Чегренець, Н. В. Руденко. – Київ : Держ. ун-т телекомунікацій, 2016. – 120 с.

39. Зубенко, В. В. Основи математичної логіки [Текст] : навч. посіб. / В. В. Зубенко, С. С. Шкільняк. – Київ : КНУБіП України, 2020. – 102 с.

40. Дичка, І. А. Комп'ютерна логіка. Прикладна теорія цифрових автоматів [Текст] : навч. посіб. / І. А. Дичка, В. П. Легеза, М. В. Онай. – Київ : Нац. техн. ун-т України «Київ. політехн. ін-т імені Ігоря Сікорського», 2018. – 88 с.

41. Фетісов, В. С. Математична система Scilab [Текст] : навч. - метод. посіб. / В. С. Фетісов. – Ніжин : НДУ ім. М. Гоголя, 2022. – 82 с.

42. Усов, А. В. Чисельні методи та їх реалізація у середовищі Scilab: [Текст] : навч. посіб. / А. В. Усов, О. А. Шпинковський, М. І. Шпинковська. – Київ : Освіта України, 2013. – 192 с.

ЗМІСТ

Навчальне видання

Даншина Світлана Юріївна

КОМП’ЮТЕРНІ ТЕХНОЛОГІЇ ДЛЯ ГІС-ДОДАТКІВ

Частина 1

ЗАГАЛЬНІ ПРИНЦИПИ ОРГАНІЗАЦІЇ КОМП’ЮТЕРНИХ ТЕХНОЛОГІЙ

Редактор А. М. Ємленінова

Зв. план, 2023
Підписано до друку 25.10.2023
Формат 60х84 1/16. Папір офс. Офс. друк
Ум. друк. арк. 5,3. Обл.-вид. арк. 6. Наклад 50 пр.
Замовлення 71-23. Ціна вільна

Видавець і виготовлювач
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»
61070, Харків-70, вул. Чкалова, 17
http:// www. khai.edu
Видавничий центр «ХАІ»
61070, Харків-70, вул. Чкалова, 17
izdat@khai.edu

Свідоцтво про внесення суб’єкта видавничої справи
до Державного реєстру видавців, виготовлювачів і розповсюджувачів
видавничої продукції сер. ДК № 391 від 30.03.2001