Тема 5. Типи і формати операндів. Способи адресації операндів

5.1 Типи і формати операндів

Машинні команди оперують даними, які в цьому випадку прийнято називати операндами. До найбільш загальних (базових) типів операндів можна віднести: адреси, числа, символи і логічні дані. Крім них ОМ забезпечує обробку і складніших інформаційних одиниць: графічних зображень, аудіо-, відео- і анімаційної інформації. Така інформація є похідною від базових типів даних і зберігається у вигляді файлів на зовнішніх запам'ятовуючих пристроях. Для кожного типу даних у ОМ передбачені певні формати.

5.1.1 Числова інформація

Серед цифрових даних можна виділити дві групи:

- цілі типи, використовувані для подання цілих чисел;
- дійсні типи для подання раціональних чисел.

В рамках першої групи є декілька форматів подання чисельної інформації, залежних від її характеру. Для подання дійсних чисел використовується форма з плаваючою комою.

Числа у формі з фіксованою комою

Подання числа X у формі з фіксованою комою (ФК), яку іноді називають також природною формою, включає знак числа і його модуль в q-ічному коді. Тут q – основа системи числення або база. Для сучасних ОМ характерна двійкова система (q = 2), але іноді використовуються також вісімкова (q = 8) або шістнадцяткова (q = 16) системи числення. Кому в записі числа називають відповідно двійковою, вісімковою або шістнадцятковою. Знак позитивного числа кодується двійковою цифрою 0, а знак негативного числа – цифрою 1.

Числам з ФК відповідає запис вигляду X = ± an-1…a1a0a-1a-2…a-r. Негативні числа зазвичай подаються в додатковому коді. Розряд коду числа, в якому розміщується знак, називається знаковим розрядом коду. Розряди, де розташовуються значущі цифри числа, називаються цифровими розрядами коду.

Знаковий розряд розміщується лівіше старшого цифрового розряду. Положення коми однакове для всіх чисел і в процесі вирішення завдань не міняється. Хоча кома і фіксується, в коді числа вона ніяк не виділяється, а тільки мається на увазі. У загальному випадку розрядна сітка ОМ для розміщення чисел у формі з ФК має вигляд, показаний на рис. 5.1, де n розрядів використовуються для запису цілої частини числа і r розрядів – для дробової частини.

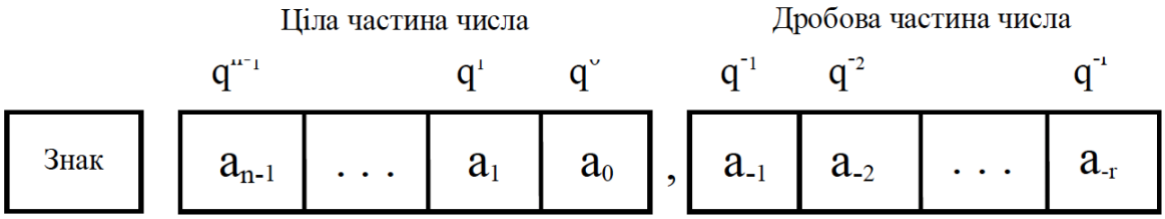

Рисунок 5.1 – Формат подання чисел з фіксованою комою

Якщо число є змішаним (містить цілу і дрібну частини), воно обробляються як ціле, хоча і не є таким (в цьому випадку застосовують термін масштабоване ціле). Обробка змішаних чисел у ОМ зустрічається украй рідко. Як правило, використовуються ОМ з дробовою(n = 0) або цілочисельною (r = 0) арифметикою.

Під час фіксації коми перед старшим цифровим розрядом можуть бути подані тільки правильні дроби.

Під час фіксації коми після молодшого розряду подаються лише цілі числа (рис. 5.2).

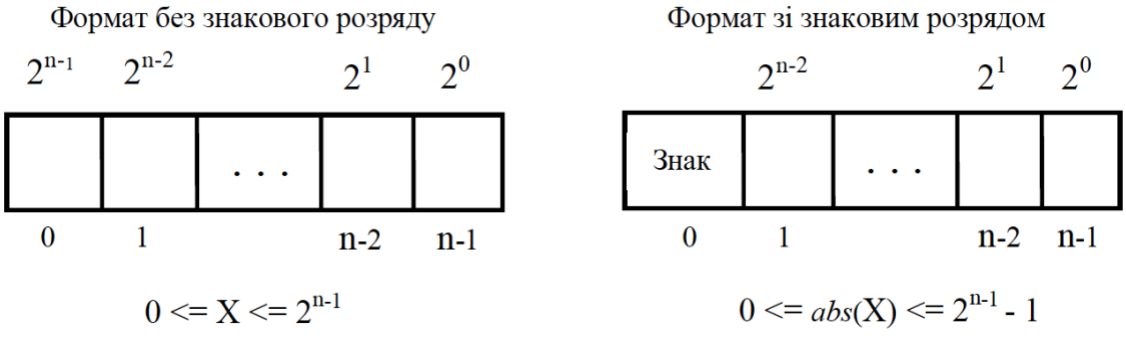

Рисунок 5.2 – Подання цілих чисел у форматі з ФК

Це найбільш поширений спосіб, тому надалі поняття ФК зв'язуватиметься виключно з цілими числами, а операції над числами у формі з ФК характеризуватимуться як цілочисельні. Тут можливі числа із знаком і без знака.

На рис. 5.2 приведені цілочисельні формати з фіксованою комою, прийняті в мікропроцесорах фірми Intel.

Подання чисел у форматі з ФК спрощує апаратурну реалізацію ОМ і скорочує час виконання машинних операцій, проте під час вирішення завдань необхідно постійно стежити за тим, щоб усі початкові дані, проміжні і остаточні результати не виходили за допустимий діапазон формату, інакше можливе переповнення розрядної сітки і результат обчислень буде невірним.

Упаковані цілі числа. В АСК сучасних мікропроцесорів є команди, що оперують цілими числами, поданими в упакованому вигляді. Зв'язано це з обробкою мультимедійної інформації. Формат припускає упаковку в межах достатньо довгого слова (зазвичай 64-розрядного) декількох невеликих цілих чисел, а відповідні команди обробляють всі ці числа паралельно. Якщо кожне з чисел складається з чотирьох двійкових розрядів, то в 64-розрядне слово мож-на помістити до 16 таких чисел. Невикористані розряди заповнюються нулями.

У мікропроцесорах фірми Intel, починаючи з Pentium MMX, присутні спеціальні команди для обробки мультимедійної інформації (MMX - команди), що оперують цілими числами, упакованими в квадрослова (рис. 5.3).

Передбачено три формати квадрослів (64-розрядних слів): упаковані байти (вісім 8-розрядних чисел); упаковані слова (чотири 16-розрядні числа) і упаковані подвійні слова (два 32-розрядні числа).

Байти у форматі упакованих байтів нумеруються від 0 до 7, причому байт 0 розташовується в молодших розрядах квадрослова. Аналогічна система нумерації і розміщення упакованих чисел застосовується для упакованих слів (номери 0–3) і упакованих подвійних слів (номери 0–1).

Ідентичні формати упакованих даних застосовуються також в іншій технології обробки мультимедійної інформації, запропонованою фірмою AMD.

Ця технологія носить назву 3DNow!, а реалізована в мікропроцесорах даної фірми.

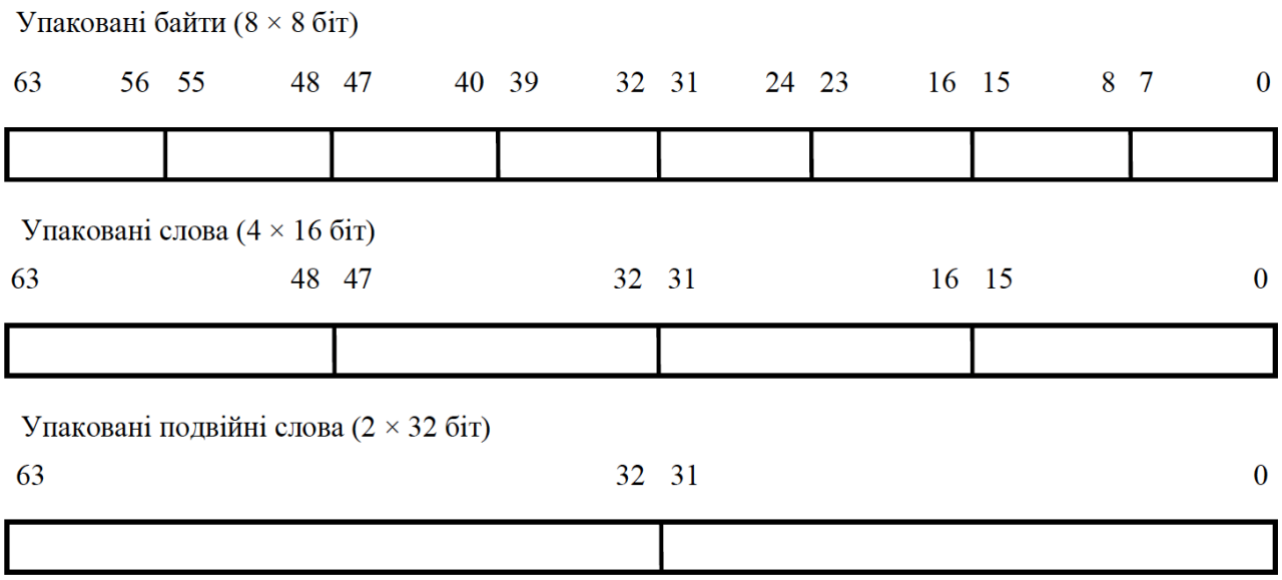

Рисунок 5.3 – Формати упакованих цілих чисел у технологіях MMX і 3DNow!

Передбачено три формати квадрослів (64-розрядних слів): упаковані байти (вісім 8-розрядних чисел); упаковані слова (чотири 16-розрядні числа) і упаковані подвійні слова (два 32-розрядні числа).

Байти у форматі упакованих байтів нумеруються від 0 до 7, причому байт 0 розташовується в молодших розрядах квадрослова. Аналогічна система нумерації і розміщення упакованих чисел застосовується для упакованих слів (номери 0–3) і упакованих подвійних слів (номери 0–1).

Ідентичні формати упакованих даних застосовуються також в іншій технології обробки мультимедійної інформації, запропонованою фірмою AMD. Ця технологія носить назву 3DNow!, а реалізована в мікропроцесорах даної фірми.

Десяткові числа. В ряді завдань, головним чином обліково-статистичного характеру, доводиться мати справу із зберіганням, обробкою і пересилкою десяткової інформації. Особливість таких завдань полягає в тому, що оброблювані числа можуть складатися з різної і дуже великої кількості

десяткових цифр. Традиційні методи обробки з перекладом початкових даних у двійкову систему числення і зворотним перетворенням результату часто зв'язані з істотними накладними витратами. З цієї причини у ОМ застосовуються інші спеціальні форми подання десяткових даних. У їх основу покладений принцип кодування кожної десяткової цифри еквівалентним двійковим числом з чотирьох бітів (тетрадою), тобто так званим двійково-десятковим кодом.

Використовуються два формати подання десяткових чисел (всі числа розглядаються як цілі): зонний (розпакований) і ущільнений (упакований). В обох форматах кожна десяткова цифра подається двійковою тетрадою, тобто замінюється двійково-десятковим кодом (рис. 5.4).

Байт		Байт			Байт		Байт	
Зона	Цифра	Зона	Цифра	. . .	Зона	Цифра	Зона	Цифра

а

Байт		Байт			Байт		Байт	
Цифра	Цифра	Цифра	Цифра	. . .	Цифра	Цифра	Цифра	Знак

б

Рисунок 5.4 – Формати десяткових чисел: а – зонний; б – ущільнений

Зонний формат (рис. 5.4, а) застосовується в операціях вводу/виводу. У ньому під кожну цифру виділяється один байт, де молодші чотири розряди відводяться під код цифри, а в старшу тетраду (поле зони) записується спеціальний код «зона», не співпадаючий з кодами цифр і знаків. Виняток становить байт, що містить молодшу цифру десяткового числа, де в полі зони зберігається знак числа.

Під час виконання операцій додавання і віднімання над десятковими числами зазвичай використовується упакований формат і в ньому ж виходить результат (множення і ділення можливе тільки в зонному форматі). В упакованому форматі (рис. 5.4, б) кожен байт містить коди двох десяткових

цифр. Права тетрада останнього байта призначається для запису знака числа. Десяткове число повинне займати цілу кількість байтів. Якщо ця умова не виконується, то чотири старші двійкові розряди лівого байта заповнюється нулями.

Розміщення знака в молодшому байті, як в зонному, так і в упакованому поданнях, дозволяє задавати десяткові числа довільної довжини і передавати їх у вигляді ланцюжка байтів. В цьому випадку знак указує, що байт, в якому він міститься, є останнім байтом даного числа, а наступний байт послідовності – це старший байт чергового числа.

Числа у формі з плаваючою комою

Від недоліків ФК в значній мірі вільна форма подання чисел з плаваючою комою (ПК), відома також під назвами нормальної або напівлогарифмічної форми. У даному варіанті кожне число розбивається на дві групи цифр. Перша група цифр називається мантисою, друга – порядком. Число подається у вигляді добутку

$$X = \pm m\, q^{\pm p}$$

де m – мантиса числа X, p – порядок числа, q – основа системи числення.

Для подання числа у формі з ПК потрібно задати знаки мантиси і порядку, їх модулі в q-ічному коді, а також основу системи числення (рис. 5.5). Нормальна форма неоднозначна, оскільки взаємна зміна m і p приводить до «плавання» коми, чим і обумовлена назва цієї форми.

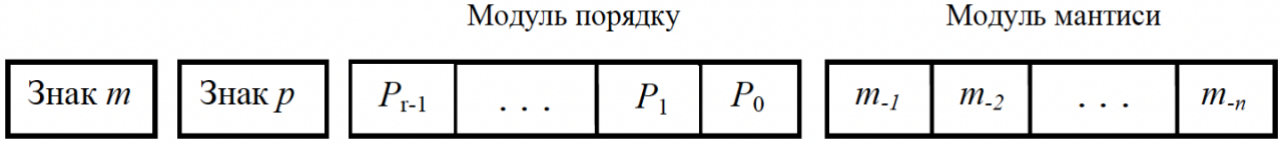

Рисунок 5.5 – Форма подання чисел з плаваючою комою

Діапазон і точність подання чисел з ПК залежать від числа розрядів, що відводяться під порядок і мантису.

У більшості обчислювальних машин для спрощення операцій над порядками останні приводять до цілих позитивних чисел, застосовуючи так званий зміщений порядок. Для цього до дійсного порядку додається ціле позитивне число – зсув (рис. 5.6). Наприклад, у системі із зсувом 128 порядок –3 подається як 125 (–3 + 128).

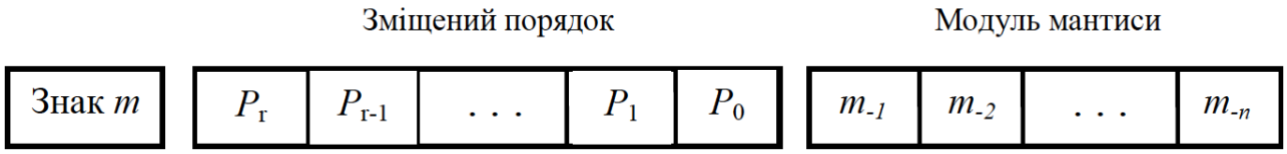

Рисунок 5.6 – Формат числа з ПК із зміщеним порядком

Звичайний зсув вибирається рівним половині уявного діапазону порядків. Відзначимо, що зміщений порядок займає всі біти поля порядку, у тому числі і той, який раніше використовувався для запису знака порядку.

Мантиса в числах з ПЗ зазвичай подається в нормалізованій формі. Це означає, що на мантису накладаються такі умови, щоб вона по модулю була менше одиниці ($|q| < 1$), а перша цифра після крапки відрізнялася від нуля. Отримана таким чином мантиса називається нормалізованою. Для двійкової системи числення можна записати: X = q 2 p, (1 > |q| = S).

Якщо перші і цифр мантиси рівні нулю, для нормалізації її потрібно зсунути відносно коми на і розрядів вліво з одночасним зменшенням порядку на і одиниць.

Якщо для запису числа з ПК використовується основа 2 (q = 2), то часто застосовують спосіб підвищення точності подання мантиси, що називається прийомом прихованої одиниці. Суть його в тому, що в нормалізованій мантисі старша цифра завжди дорівнює одиниці (для подання нуля використовується спеціальна кодова комбінація), отже, ця цифра може не записуватися, а матися на увазі. Запис мантиси починають з її другої цифри, і це дозволяє задіювати додатковий значущий біт для точнішого подання числа. Слід зазначити, що значення порядку в даному випадку не міняється. Прихована одиниця перед

виконанням арифметичних операцій відновлюється, а під час запису результату – віддаляється.

Для істотнішого збільшення точності обчислень під число відводять декілька машинних слів, наприклад два. Додаткові біти, як правило, служать для збільшення розрядності мантиси, проте у ряді випадків частина з них може відводитися і для розширення поля порядку. В процесі обчислень може виходити ненормалізоване число. У такому разі ОМ, якщо це наказано командою, автоматично нормалізує його.

Числа з плаваючою комою в різних ОМ мають декілька різних форматів. У даний час для всіх ОМ рекомендований стандарт, розроблений загальновизнаним міжнародним центром стандартизації IEEE (Institute of Electrical and Electronics Engineers) - IEEE 754 з метою полегшення переносу програм з одного процесора на інші.

Стандарт визначає 32-бітовий (одинарний) і 64-бітовий (подвійний) формати з 8- і 11-розрядним порядком відповідно. Основою системи числення є 2. На додаток стандарт передбачає два розширені формати, одинарний і подвійний, фактичний вид яких залежить від конкретної реалізації. Розширені формати передбачають додаткові біти для порядку (збільшений діапазон) і мантиси (підвищена точність).

В останніх версіях АСК, що передбачають особливі команди для обробки мультимедійної інформації, крім упакованих цілих чисел використовуються і упаковані числа з плаваючою комою. Так, у вже згадуваній технології 3DNow! фірми AMD є команди, які служать для збільшення продуктивності систем під час обробки тривимірних застосувань, що описуються числами з ПК. Кожна така команда працює з двома операндами з плаваючою комою одинарної точності, які упаковуються в 64-розрядні групи.

У мікропроцесорах фірми Intel, починаючи з Pentium III, для аналогічних цілей введені команди, що реалізовують технологію SSE (Streaming SIMD Extension – потокова обробка за принципом «одна команда – багато даних»), також орієнтовану на паралельну обробку упакованих чисел з ПК. Тут числа

об'єднуються в групи завдовжки 128 біт, і це дозволяє упакувати в групу чотири 32-розрядні числа з ПК (числа з одинарною точністю). Пізніше, в технології SSE2, яку можна вважати подальшим розвитком SSE, з'явився формат, де в групу з 128 біт упаковуються два 64-розрядні числа з ПК, тобто числа, подані з подвійною точністю.

Розрядність основних форматів числових даних і розміщення їх у пам'яті

Дані, що подають у ОМ числову інформацію, можуть мати фіксовану або змінну довжину. Операційні пристрої обчислювальних машин (цілочисельні арифметико-логічні пристрої, блоки обробки чисел з плаваючою комою, пристрою десяткової арифметики і т. п.), як правило, розраховані на обробку коду фіксованої довжини.

Найменшою одиницею даних у ОМ служить біт (BIT, BInary digiT - двійкова цифра). В більшості випадків ця одиниця інформації дуже мала. Однобітові операційні пристрої використовувалися в ОМ з послідовною обробкою інформації, а в сучасних машинах з паралельною обробкою розрядів вони практично не застосовуються. Побітову роботу з даними швидше можна зустріти в багатопроцесорних обчислювальних системах, побудованих з однорозрядних процесорів.

Реально найменшою оброблюваною одиницею вважається байт, що складається з восьми бітів. На практиці ця одиниця інформації також виявляється недостатньою, і значно частіше застосовуються числа, подані двома (півслово), чотирма (слово), вісьма (подвійне слово) або шістнадцятьма (зчетверене слово) байтами.

Розрядність цілочисельного АЛП зазвичай вибирається рівною ширині адреси (для більшості сучасних ОМ це 32 розряди). Отже, найбільш вигідними в плані швидкодії є такі цілі числа, довжина яких збігається з розрядністю адреси. Використання коротших чисел дозволяє заощадити на пам'яті, але виграшу в продуктивності не дає.

Блоки операцій з плаваючою комою зазвичай узгоджені із стандартом IEEE 754 і розраховані на обробку чисел у форматі подвійної довжини (64 біти). В більшості ОМ реальна розрядність таких блоків навіть більше (80 біт). Таким чином, якнайкращим варіантом під час проведення обчислень з плаваючою комою можна вважати формат подвійного слова. Під час вибору формату меншої довжини (32 розряди) обчислення все одно ведуться з більшою точністю, після чого результат округляється. Таким чином, використання короткого формату чисел з плаваючою комою, як і у разі цілих чисел з фіксованою комою, крім економії пам'яті ніяких інших переваг також не дає.

В сучасних ОМ розрядність однієї комірки пам'яті, як правило, рівна одному байту (8 біт). В той же час реальна довжина кодів чисел складає 2, 4, 8 або 16 байт. У разі зберігання таких чисел у пам'яті послідовні байти числа розміщують в декількох комірках з послідовними адресами, при цьому для доступу до числа указується тільки найменша з адрес. Під час розробки архітектури системи команд необхідно визначити порядок розміщення байтів у пам'яті, тобто якому з байтів (старшому або молодшому) відповідатиме ця найменша адреса.

В обчислювальному плані обидва способи запису рівноцінні. Так, фірми DEC і Intel віддають переваги розміщенню в першій комірці молодшого байта, а IBM і Motorola орієнтуються на протилежний варіант. Вибір зазвичай пов'язаний з якимись іншими міркуваннями розробників ОМ. В даний час в більшості машин передбачається використання обох варіантів. Крім порядку розміщення байтів, істотним буває і вибір адреси, з якої може починатися запис числа. Зв'язано це з фізичною реалізацією напівпровідникових запам'ятовуючих пристроїв, де зазвичай передбачається можливість читання (запису) чотирьох байтів підряд. Причому дана операція виконується швидше, якщо адреса першого байта A відповідає умові A mod S= 0, (S = 2, 4, 8, 16). Числа, розміщені в пам'яті відповідно до цього правила, називаються такими, що вирівнюються.

5.1.2 Інші види інформації

Символьна інформація. У загальному об'ємі обчислювальних дій все більша частка приходиться на обробку символьної інформації, що містить букви, цифри, розділові знаки, математичні та інші символи. Кожному символу ставиться у відповідність певна двійкова комбінація. Сукупність можливих символів і призначених їм двійкових кодів утворює таблицю кодування. В даний час застосовується багато різних таблиць кодування. Об'єднує їх ваговий принцип, за якого ваги кодів цифр зростають у міру збільшення цифри, а ваги символів збільшуються в алфавітному порядку. Так, вага букви «Б» на одиницю більше ваги букви «А». Це сприяє спрощенню обробки в ОМ. Найбільш поширеними є кодові таблиці, в яких символи кодуються за допомогою восьмирозрядних двійкових комбінацій (байтів), що дозволяють подати 256 різних символів:

- розширений двійково-кодований код EBCDIC (Extended Binary Coded Decimal Interchange Code);
- американський стандартний код для обміну інформацією ASCII (American Standard Code for Information Interchange).

Код EBCDIC використовується як внутрішній код в універсальних ОМ фірми IBM. Він же відомий під назвою ДКОІ (двійковий код для обробки інформації).

Стандартний код ASCII – 7-розрядний, восьма позиція відводиться для запису біта парності. Це забезпечує уявлення 128 символів, включаючи всі латинські букви, цифри, знаки основних математичних операцій і знаки пунктуації. Пізніше з'явилася європейська модифікація ASCII, звана Latin 1 (стандарт ISO 8859-1). У ній «корисно» використовуються всі 8 розрядів. Додаткові комбінації (коди 128-255) в новому варіанті відводяться для подання специфічних букв алфавітів західноєвропейських мов, символів псевдографіки, деяких букв грецького алфавіту, а також ряду математичних і фінансових символів. Саме ця кодова таблиця вважається світовим стандартом де-факто, який застосовується з різними модифікаціями у всіх країнах.

Хоча код ASCII достатньо зручний, він все ж таки дуже тісний і не вміщає багатьох необхідних символів. З цієї причини в 1993 році консорціумом компаній Apple Computer, Microsoft, Hewlett-Packard, DEC і IBM був розроблений 16-бітовий стандарт ISO 10646, що визначає універсальний набір символів (UCS, Universal Character Set). Новий код, відомий під назвою Unicode, дозволяє задати до 65 536 символів, тобто дає можливість одночасно подавати символи всіх основних «живих» і «мертвих» мов. Для букв російської мови виділені коди 1040-1093.

Логічні дані. Елементом логічних даних є логічна (булева) змінна, яка може приймати лише два значення: «істина» або «неправда». Кодування логічного значення прийнято здійснювати бітом інформації: одиницею кодують дійсне значення, нулем – помилкове. Як правило, в ОМ оперують наборами логічних змінних завдовжки в машинне слово. Обробляються такі слова за допомогою команд логічних операцій (І, АБО, НІ і т. д.), при цьому всі біти обробляються однаково, але незалежно один від одного, тобто ніяких перенесень між розрядами не виникає.

Рядки. Рядки – це безперервна послідовність бітів, байтів, слів або подвійних слів. Бітовий рядок може починатися в будь-якої позиції байта і містити до 232 біт. Байтовий рядок може складатися з байтів, слів або подвійних слів. Довжина такого рядка варіюється від нуля до 232-1 байт (4 Гбайт). Приведені цифри характерні для переважаючих у даний час 32-розрядних ОМ.

Якщо байтами байтового рядка є коди символів, то говорять про текстовий рядок. Оскільки довжина текстового рядка може мінятися в дуже широких межах, то для вказівки кінця рядка в останній байт заноситься код-обмежувач – звичайно це нулі у всіх розрядах байта. Іноді замість обмежувача довжину рядка указують числом, розташованим в першому байті (двох) рядка.

Відеоінформація. Відеоінформація буває як статичною, так і динамічною. Статична відеоінформація включає текст, малюнки, графіки, креслення, таблиці та ін. Малюнки діляться також на плоскі – двовимірні і об'ємні – тривимірні. Динамічна відеоінформація – це відео-, мульт- і слайд-фільми. В їх основі лежить

послідовне експонування на екрані в реальному масштабі часу окремих кадрів відповідно до сценарію. Динамічна інформація використовується або для передачі рухомих зображень (анімація), або для послідовної демонстрації окремих кадрів (слайд-фільми).

В обчислювальній техніці існує два способи подання графічних зображень: матричний (растровий) і векторний. Матричні (bitmap) формати добре підходять для зображень із складною гамою кольорів, відтінків і форм, таких як фотографії, малюнки, відскановані дані. Векторні формати більш пристосовані для креслень і зображень з простими формами, тінями і забарвленням.

У матричних форматах зображення подається прямокутною матрицею точок – пікселів (picture element), положення яких у матриці відповідає координатам точок на екрані. Крім координат кожен піксел характеризується своїм кольором, кольором фону або градацією яскравості. Кількість бітів, що виділяються для вказівки кольору піксела, змінюється залежно від формату. У високоякісних зображеннях колір піксела описують 24 бітами, що дає близько 16 млн. кольорів. Основний недолік матричної (растрової) графіки полягає у великій ємності пам'яті, потрібної для зберігання зображення. У даний час існує множина форматів графічних файлів, що розрізняються алгоритмами стиснення і способами подання матричних зображень, а також сферою застосування.

Векторне подання, на відміну від матричної графіки, визначає опис зображення не пікселами, а кривими – сплайнами. Сплайн– це гладка крива, яка проходить через дві або більше опорні точки, що керують формою сплайна.

Основні переваги векторної графіки:

- опис об'єкта є простим і займає мало пам'яті;
- простота масштабування зображення без погіршення його якості;
- незалежність ємності пам'яті, потрібної для зберігання зображення, від вибраної колірної моделі.

Недоліком векторних зображень є їх деяка штучність, що полягає в тому, що будь-яке зображення необхідно розбити на кінцеву множину складових його

примітивів. Примітив – ряд простих об'єктів, наприклад еліпс, прямокутник, лінія, які служать основою для створення складніших зображень.

Матрична і векторна графіка існують не відособлено одна від одної. Так, векторні малюнки можуть включати і матричні зображення. Крім того, векторні і матричні зображення можуть бути перетворені один в одного. Графічні формати, що дозволяють поєднувати матричний і векторний опис зображення, називаються метафайлами. Метафайли забезпечують достатню компактність файлів із збереженням високої якості зображення.

Аудіоінформація. Поняття аудіо пов'язане із звуками, які здатне сприймати людське вухо. Частоти аудіосигналів лежать в діапазоні від 15 Гц до 20 кГц, а сигнали за своєю природою є безперервними (аналоговими). Перш ніж бути поданою у ОМ, аудіоінформація повинна бути перетворена в цифрову форму (оцифрована). Для цього значення звукових сигналів (вибірки, samples), узяті через малі проміжки часу, за допомогою аналого-цифрових перетворювачів (АЦП) переводяться в двійковий код. Зворотна дія виконується цифро-аналоговими перетворювачами (ЦАП). Чим частіше проводяться вибірки, тим вище може бути точність подальшого відтворення початкового сигналу, але тим більша ємність пам'яті потрібна для зберігання оцифрованого звуку.

5.2 Способи адресації операндів

Питання про те, яким чином в адресному полі команди може бути вказане місцеположення операндів, вважається одним з центральних при розробці архітектури ОМ. З погляду скорочення апаратурних витрат очевидне прагнення розробників зменшити довжину адресного поля при збереженні можливостей доступу до всього адресного простору. З другого боку, спосіб завдання адрес повинен сприяти максимальному зближенню конструктив мов програмування високого рівня і машинних команд. Все це привело до того, що в архітектурі системи команд будь-якої ОМ передбачені різні способи адресації операндів.

Приступаючи до розгляду способів адресації, спочатку визначимо поняття «виконавчий» і «адресний код».

Виконавчою адресою операнда (Ав) називається двійковий код номера комірки пам'яті, яка служить джерелом або приймачем операнда. Цей код подається на адресні входи запам'ятовуючого пристрою (ЗП), і по ньому відбувається фактичне звернення до вказаної комірки. Якщо операнд зберігається не в основній пам'яті, а в регістрі процесора, його виконавчою адресою буде номер регістра.

Адресний код команди (Ак) – це двійковий код в адресному полі команди, з якого необхідно сформувати виконавчу адресу операнда.

У сучасних ОМ виконавча адреса і адресний код, як правило, не збігаються, і для доступу до даних потрібне відповідне перетворення. Спосіб адресації – це спосіб формування виконавчої адреси операнда за адресним кодом команди. Спосіб адресації істотно впливає на параметри процесу обробки інформації. Одні способи дозволяють збільшити ємність пам'яті, що адресується, без подовження команди, але знижують швидкість виконання операції, інші – прискорюють операції над масивами даних, треті – спрощують роботу з підпрограмами і так далі. В сьогоднішніх ОМ зазвичай є можливість застосування декількох різних способів адресації операндів до однієї і тієї ж операції.

Щоб пристрій управління обчислювальної машини міг визначити, який саме спосіб адресації прийнятий у даній команді, в різних ОМ використовуються різні прийоми. Часто різним способам адресації відповідають і різні коди операції. Інший підхід – це додавання до складу команди спеціального поля способу адресації, вміст якого визначає, який із способів адресації повинен бути застосований. Іноді в команді є декілька полів – поодинці на кожну адресу. Відзначимо, що можливий також варіант, коли в команді взагалі відсутня адресна інформація, тобто має місце неявна адресація. За неявної адресації адресного поля або взагалі немає, або воно містить не всі необхідні адреси – відсутня адреса мається на увазі кодом операції.

В даний час використовуються різні види адресації, найбільш поширені з яких розглядаються нижче.

Безпосередня адресація. Під час безпосередньої адресації (БА) в адресному полі команди замість адреси міститься безпосередньо сам операнд (рис. 5.7). Цей спосіб може застосовуватися під час виконання арифметичних операцій, операцій порівняння, а також для завантаження констант у регістри. Коли операндом є число, воно зазвичай подається в додатковому коді.

Під час запису в регістр, в якому розрядність перевищує довжину безпосереднього операнда, операнд розміщується в молодшій частині регістра, а позиції, що залишилися вільними, заповнюються значенням знакового біта операнда.

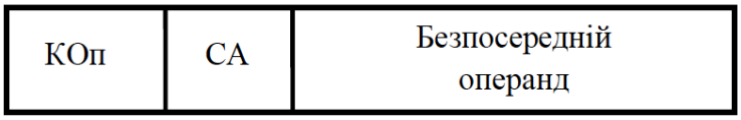

Рисунок 5.7 – Безпосередня адресація

Крім того, що в адресному полі можуть бути вказані тільки константи, ще одним недоліком даного способу адресації є те, що розмір безпосереднього операнда обмежений довжиною адресного поля команди, яке в більшості випадків менше довжини машинного слова. Безпосередня адресація скорочує час виконання команди, оскільки не потрібне звернення до пам'яті за операндом. Крім того, економиться пам'ять, оскільки відпадає необхідність в комірці для зберігання операнда.

Пряма адресація. У випадку прямої або абсолютної адресації (ПА) адресний код прямо указує номер комірки пам'яті, до якої проводиться звернення (рис. 5.8).

При всій простоті використання спосіб має істотний недолік – обмежений розмір адресного простору, оскільки для адресації до пам'яті великої ємності потрібне «довге» адресне поле. Проте істотнішою недосконалістю можна вважати те, що адреса, вказана в команді, не може бути зміненою в процесі обчислень (в усякому разі, така зміна не рекомендується). Це обмежує можливості щодо довільного розміщення програми в пам'яті.

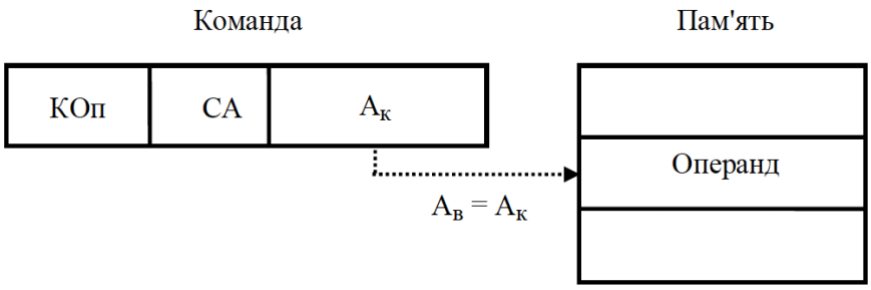

Рисунок 5.8. Пряма адресація

При всій простоті використання спосіб має істотний недолік – обмежений розмір адресного простору, оскільки для адресації до пам'яті великої ємності потрібне «довге» адресне поле. Проте істотнішою недосконалістю можна вважати те, що адреса, вказана в команді, не може бути зміненою в процесі обчислень (в усякому разі, така зміна не рекомендується). Це обмежує можливості щодо довільного розміщення програми в пам'яті.

Непряма адресація. Одним із шляхів подолання проблем, властивих прямій адресації, може служити прийом, коли за допомогою обмеженого адресного поля команди вказується адреса комірки, що у свою чергу містить повнорозрядну адресу операнда (рис. 5.9). Цей спосіб відомий як непряма адресація (НА). Запис (Ак) означає вміст комірки, адреса якої вказана в дужках.

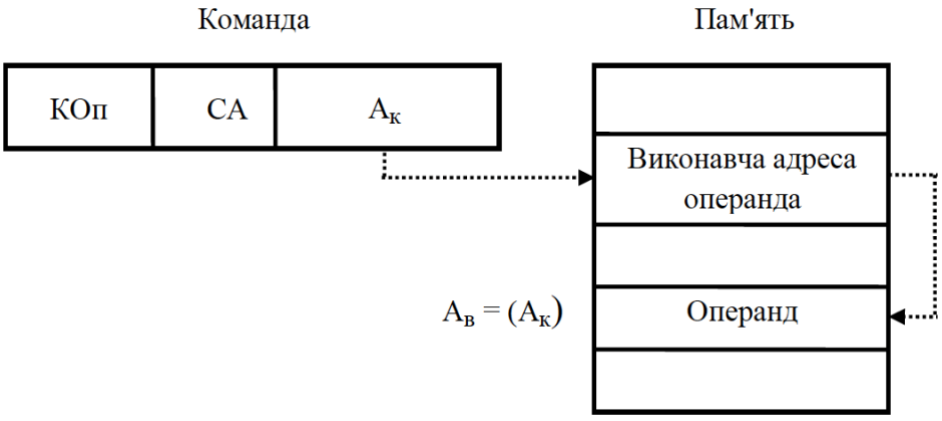

Рисунок 5.9 – Непряма адресація

У випадку непрямої адресації вміст адресного поля команди залишається незмінним, тоді як непряму адресу в процесі виконання програми можна

змінювати. Це дозволяє проводити обчислення, коли адреси операндів заздалегідь невідомі і з'являються лише в процесі розв'язання задачі. Додатково такий прийом спрощує обробку масивів і списків, а також передачу параметрів підпрограмам.

Недоліком непрямої адресації є необхідність у двократному зверненні до пам'яті: спочатку для витягання адреси операнда, а потім для звернення до операнда. Понад те задіюється зайва комірка пам'яті для зберігання виконавчої адреси операнда.

Регістрова адресація (РА) нагадує пряму адресацію. Відмінність полягає в тому, що адресне поле інструкції указує не на комірку пам'яті, а на регістр процесора (рис. 5.10). Ідентифікатор регістра надалі позначатимемо буквою R. Зазвичай розмір адресного поля в даному випадку складає три або чотири біти, що дозволяє вказати відповідно на один з 8 або 16 регістрів загального призначення (РЗП).

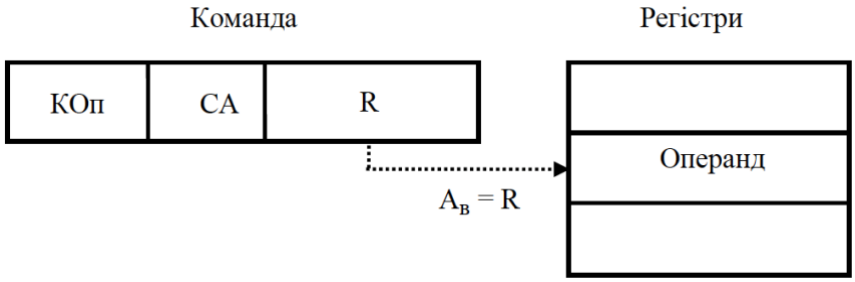

Рисунок 5.10 – Регістрова адресація

Двома основними перевагами регістрової адресації є: коротке адресне поле в команді і виключення звернень до пам'яті.

На жаль, можливості по використанню регістрової адресації обмежені малим числом РЗП у складі процесора.

Непряма регістрова адресація (НРА) є непрямою адресацією, де виконавча адреса операнда зберігається не в комірці основної пам'яті, а в регістрі процесора. Відповідно, адресне поле команди указує не на комірку пам'яті, а на регістр (рис. 5.11).

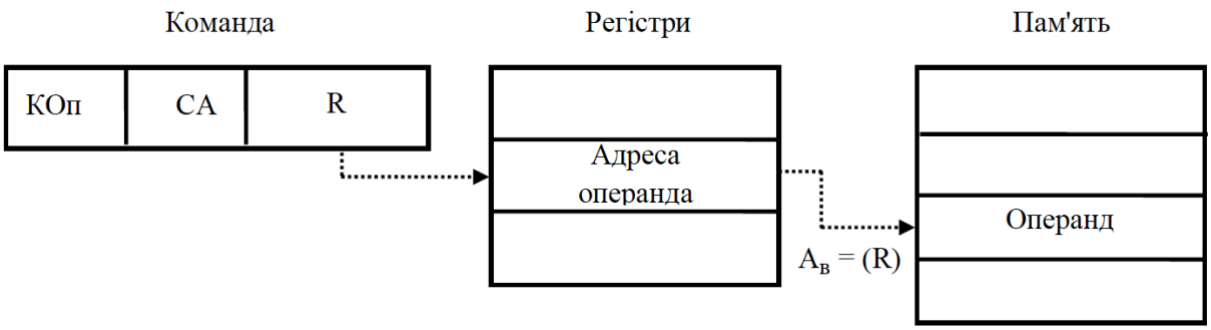

Рисунок 5.11 – Непряма регістрова адресація

Переваги і обмеження непрямої регістрової адресації ті ж, що і у звичайної непрямої адресації, але завдяки тому, що непряма адреса зберігається не в пам'яті, а в регістрі, для доступу до операнда потрібно на одне звернення до пам'яті менше.

Адресація зі зміщенням. Під час адресації зі зміщенням виконавча адреса формується в результаті підсумовування вмісту адресного поля команди з вмістом одного або декількох регістрів процесора (рис. 5.12). Адресація зі зміщенням припускає, що адресна частина команди включає як мінімум одне поле (Ак). В ньому міститься константа, сенс якої в різних варіантах адресації зі зміщенням може мінятися. Константа може бути якоюсь базовою адресою, до якої додається зміщення, що зберігається в регістрі.

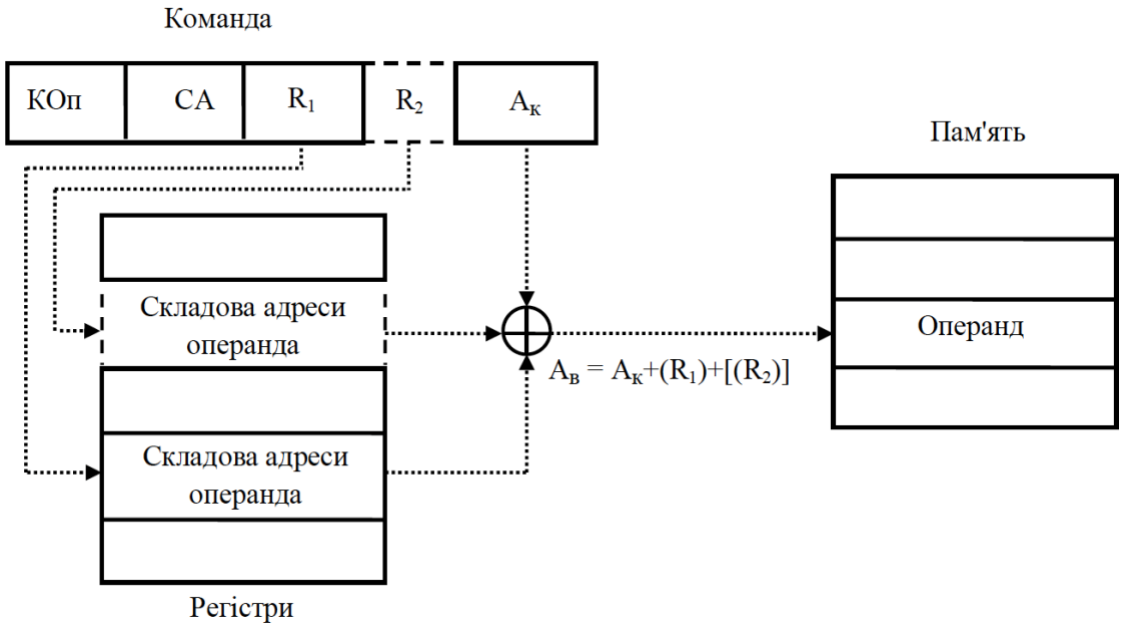

Рисунок 5.12 – Адресація зі зміщенням

Допустимий і прямо протилежний підхід: базова адреса знаходиться в регістрі процесора, а в полі Ак указується зміщення щодо цієї адреси.

В деяких процесорах для реалізації певних варіантів адресації зі зміщенням передбачені спеціальні регістри, наприклад базовий або індексний.

Якщо ж складова адреси може розташовуватися в довільному регістрі загального призначення, то для вказівки конкретного регістра в команду включається додаткове поле R (у випадку складання адреси більш ніж з двох складових у команді буде декілька таких полів). У найбільш загальному випадку адресація зі зміщенням має на увазі наявність двох адресних полів: Ак і R.

У рамках адресації зі зміщенням є ще один варіант, при якому виконавча адреса обчислюється не підсумовуванням, а конкатенацією (приєднанням) складових адреси.

Нижче розглядаються основні способи адресації зі зміщенням, кожен з яких має власну назву.

Відносна адресація. У випадку відносної адресації (ВА) для отримання виконавчої адреси операнда вміст підполя Ак команди складається з вмістом лічильника команд (рис. 5.13). Таким чином, адресний код у команді є зміщенням щодо адреси поточної команди.

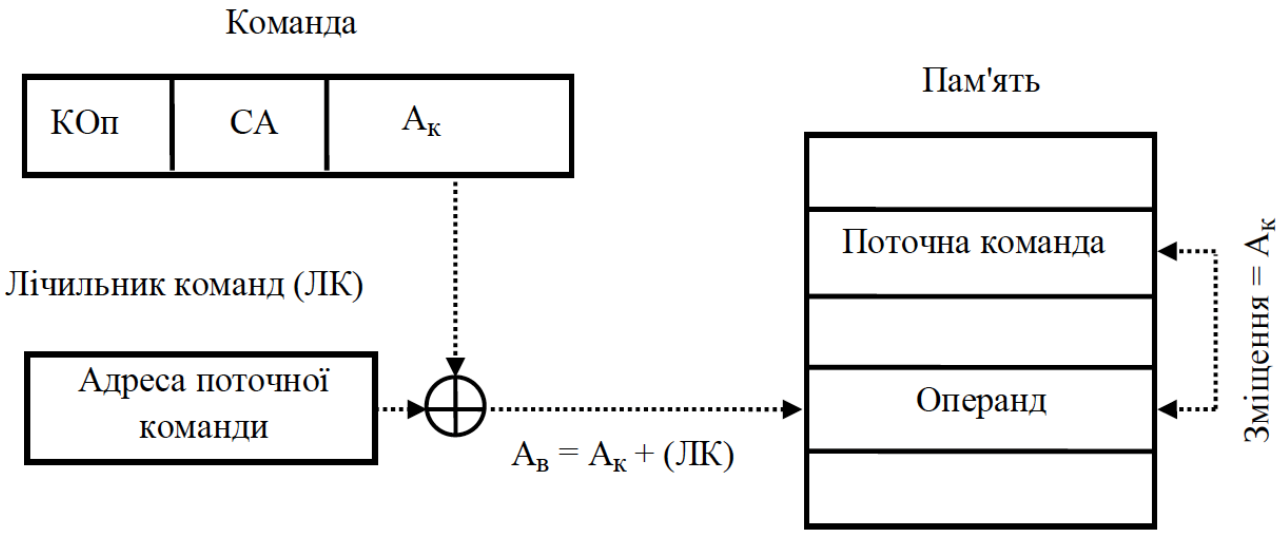

Рисунок 5.13 – Відносна адресація

Слід зазначити, що у момент обчислення виконавчої адреси операнда в лічильнику команд може вже бути сформована адреса наступної команди, що потрібно враховувати під час вибору величини зміщення. Зазвичай підполе Ак трактується як двійкове число в додатковому коді.

Адресація відносно лічильника команд базується на властивості локальності, що виражається в тому, що більша частина звернень відбувається до комірок, розташованих у безпосередній близькості від виконуваної команди. Це дозволяє заощадити на довжині адресної частини команди, оскільки розрядність підполя Ак може бути невеликою. Головна перевага даного способу адресації полягає в тому, що він робить програму переміщуваною в пам'яті: незалежно від поточного розташування програми в адресному просторі взаємне положення команди і операнда залишається незмінним, тому адресація операнда залишається коректною.

Базова регістрова адресація. У разі базової регістрової адресації (БРА) регістр, званий базовим, містить повнорозрядну адресу, а підполе Аз - зміщення щодо цієї адреси (рис. 5.14).

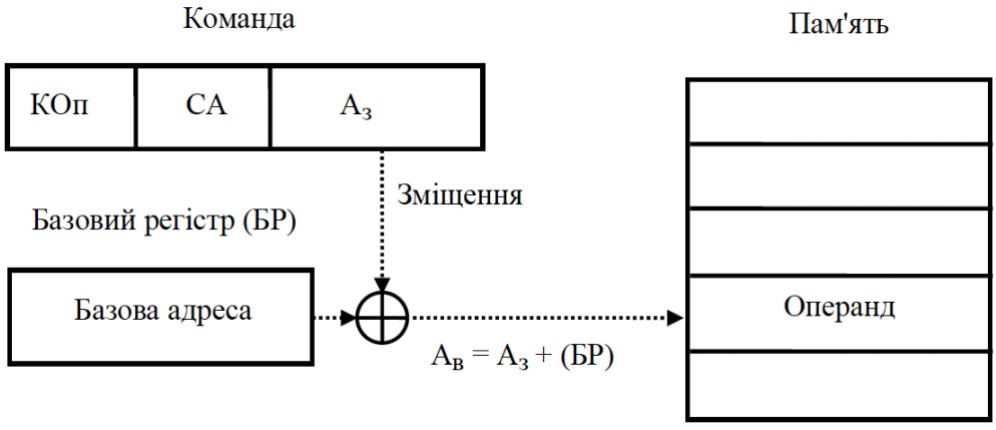

Рисунок 5.14 – Базова регістрова адресація з базовим регістром

Посилання на базовий регістр може бути явним або неявним. У деяких ОМ є спеціальний базовий регістр і його використання є неявним, тобто підполе R в команді відсутнє

Типовіший випадок, коли в ролі базового регістра виступає один з регістрів загального призначення (РЗП), тоді його номер явно вказується в підполі R команди (рис. 5.15).

Базову регістрову адресацію зазвичай використовують для доступу до елементів масиву, положення якого в пам'яті в процесі обчислень може мінятися. У базовий регістр заноситься початкова адреса масиву, а адреса елемента масиву вказується в підполі Аз команди у вигляді зміщення щодо початкової адреси масиву.

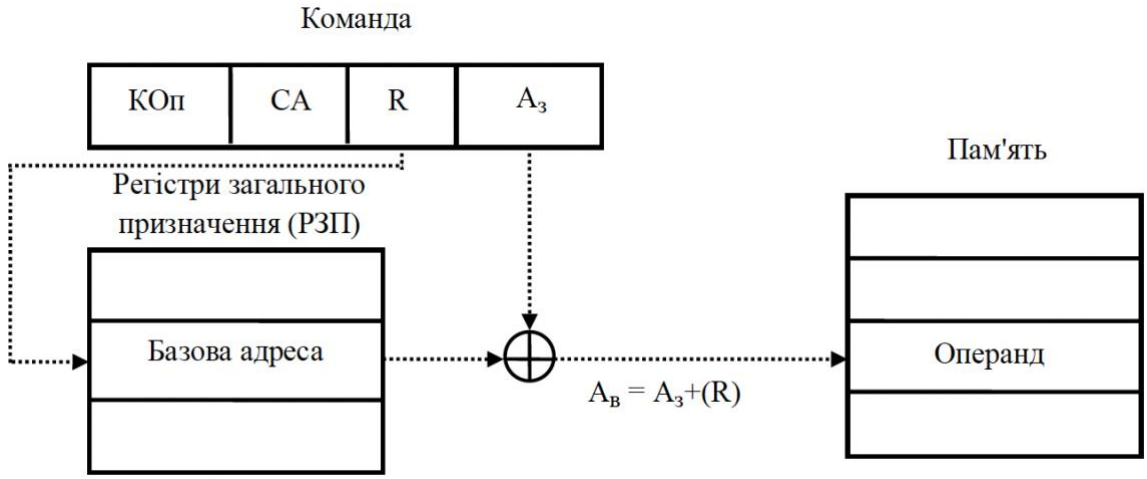

Рисунок 5.15 – Базова регістрова адресація з використанням одного з РЗП

Перевага даного способу адресації в тому, що зміщення має меншу довжину, чим повна адреса, і це дозволяє скоротити довжину адресного поля команди.

Індексна адресація. У разі індексної адресації (ІА) підполе Аб містить адресу комірки пам'яті, а регістр (вказаний явно або неявно) – зміщення щодо цієї адреси (рис. 5.16).

Як видно, цей спосіб адресації схожий на базову регістрову адресацію. Оскільки у разі індексної адресації в полі Аб знаходиться повнорозрядна адреса комірки пам'яті, що грає роль бази, довжина цього поля більша, ніж у разі базової регістрової адресації. Проте обчислення виконавчої адреси операнда проводиться ідентично.

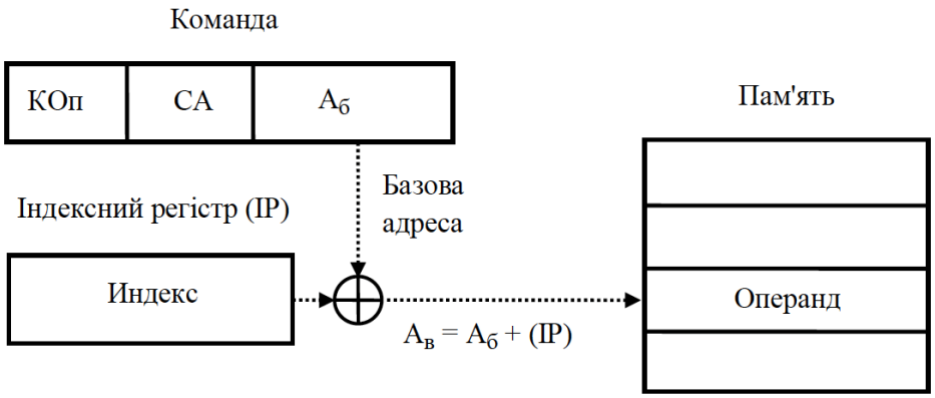

Рисунок 5.16 – Індексна адресація з індексним регістром

У разі, коли в ролі індексного регістра для зберігання зміщення виступає один з регістрів загального призначення (РЗП), тоді його номер явно вказується в підполі R команди (рис. 5.17).

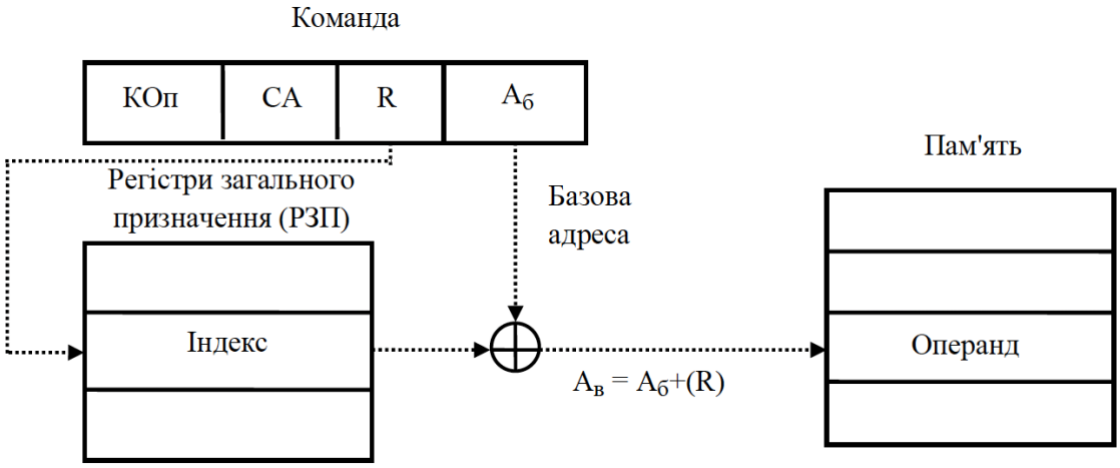

Рисунок 5.17 – Індексна адресація з використанням одного з РЗП

Оскільки це досить типовий випадок, у більшості ОМ збільшення або зменшення вмісту індексного регістра до або після звернення до нього здійснюється автоматично як частина машинного циклу. Такий прийом називається автоіндексуванням. Якщо для індексної адресації використовуються спеціально виділені регістри, автоіндексування може проводитися неявно і автоматично. У випадку задіювання для зберігання індексів регістрів загального призначення необхідність операції автоіндексування повинна указуватися в команді спеціальним бітом.

Автоіндексування із збільшенням вмісту індексного регістра носить назву автоінкрементної адресації і може бути описано таким чином:

$$A_в = A_б + (\mathrm{R}),\ R < (\mathrm{R}) + 1 \text{ або } R < (\mathrm{R}) + 1,\ A_в = A_б + (\mathrm{R})$$

У першому варіанті збільшення вмісту індексного регістра відбувається після формування виконавчої адреси, і цей спосіб називається постінкрементним автоіндексуванням. У другому випадку спочатку проводиться збільшення вмісту індексного регістра, і вже нове значення використовується для формування виконавчої адреси. Тоді говорять про преінкрементне автоіндексування.

Автоіндексування із зменшенням вмісту індексного регістра носить назву автодекрементної адресації і може бути описано так:

$$A_в = A_б + (\mathrm{R}),\ R < (\mathrm{R}) - 1 \text{ або } R < (\mathrm{R}) - 1,\ A_в = A_б + (\mathrm{R})$$

Тут також можливі два варіанти, що відрізняються послідовністю виконання операцій зменшення вмісту індексного регістра і обчислення виконавчої адреси: постдекрементне автоіндексування і переддекрементне автоіндексування.

Існує ще один варіант індексної адресації – індексна адресація з масштабуванням і зсувом: вміст індексного регістра множиться на масштабний коефіцієнт і підсумовується з Аб. Масштабний коефіцієнт може приймати значення 1, 2, 4 або 8, для чого в адресній частині команди виділяється додаткове поле.

Сторінкова адресація (СТА) припускає розбиття адресного простору на сторінки. Сторінка визначається своєю початковою адресою, що виступає як база. Старша частина цієї адреси зберігається в спеціальному регістрі – регістрі адреси сторінки (РАС). В адресному коді команди вказується зміщення усередині сторінки, що розглядається як молодша частина виконавчої адреси. Виконавча адреса утворюється конкатенацією (приєднанням, символ ||) Аз до вмісту РАС, як показано на рис. 5.18.

Блочна адресація використовується в командах, для яких одиницею обробки служить блок даних, розташованих у послідовних комірках пам'яті. Цей спосіб дуже зручний під час роботи із зовнішніми запам'ятовуючими пристроями і в операціях з векторами. Для опису блока зазвичай береться адреса комірки, де

зберігається перший або останній елемент блока, і загальна кількість елементів блока, яка задана числом байтів або комірок. Замість довжини блока може використовуватися спеціальна ознака «кінець блока», що поміщається за останнім елементом блока

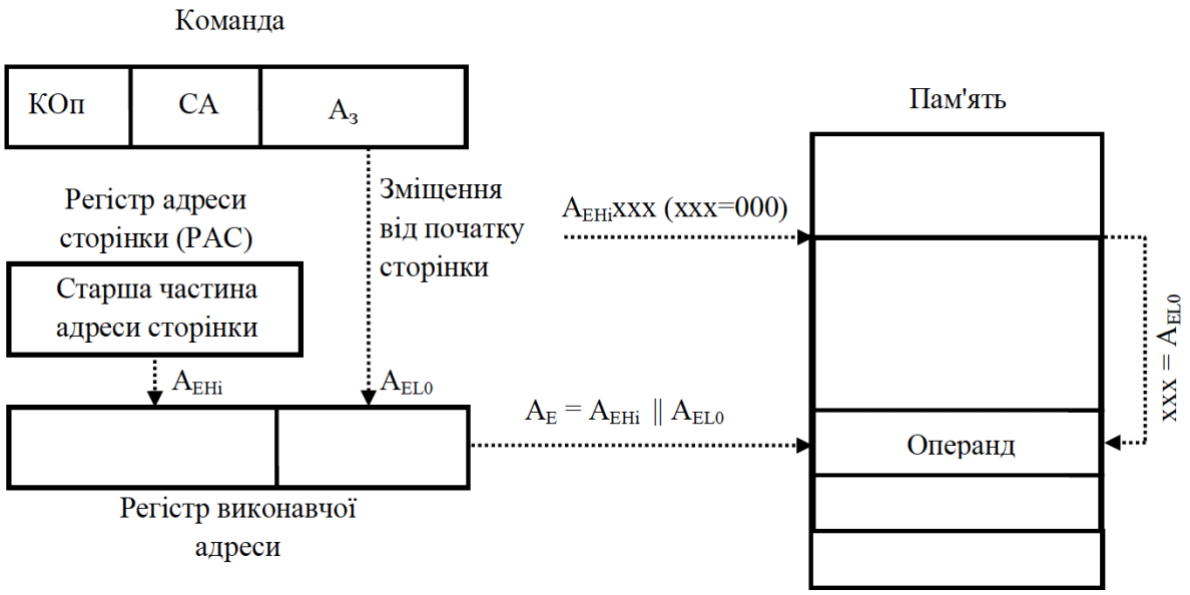

Рисунок 5.18 – Сторінкова адресація

Стекова адресація буде розглянута під час опису стекової архітектури системи команд.