23 АРХІТЕКТУРА МІКРОПРОЦЕСОРІВ

23.1 Типи процесорів

З погляду користувача при виборі мікропроцесора доцільно мати у своєму розпорядженні деякі узагальнені комплексні характеристики можливостей мікропроцесора. Розроблювач має потребу в з'ясуванні й розумінні лише тих компонентів мікропроцесора, які явно відбиваються в програмах і повинні бути враховані при розробці схем і програм функціонування системи. Такі характеристики визначаються поняттям архітектури мікропроцесора.

Архітектура мікропроцесора – це його логічна організація, розглянута з погляду користувача; вона визначає можливості мікропроцесора за апаратною і програмною реалізаціями функцій, необхідних для побудови мікропроцесорної системи. Поняття архітектури мікропроцесора відображає:

- його структуру, тобто сукупність компонентів, що становлять мікропроцесор, і зв'язків між ними; для користувача досить обмежитися регістровою моделлю мікропроцесора;

- способи подання й формати даних;

- способи доступу до всіх програмно-доступних для користувача елементів структури (адресація до регістрів, комірок постійної й оперативної пам'яті, зовнішніх пристроїв);

- набір операцій, виконуваних мікропроцесором;

- характеристики керуючих слів і сигналів, вироблюваних мікропроцесором, і тих, які входять до нього ззовні;

- реакцію на зовнішні сигнали (система обробки переривань тощо).

За організацією процесу обчислення і зберігання проміжних результатів можна виділити наступні типи процесорів.

Акумуляторна архітектура – з регістрів виділяється один з кілька регістрів-акумуляторів. Регістр-акумулятор є джерелом одного з аргументів і приймачем результату обчислень. Операції кодуються як правило в одноопернадні інструкції (акумулятор + операнд → акумулятор). Така архітектура характерна для багатьох CISC-процесорів.

Стекова архітектура - визначається організацією реєстрового файлу в вигляді стека, і непрямою адресацією регістрів через покажчик стека, який визначає положення вершини стека, операції проводяться над значеннями на вершині стека і результат кладеться також на вершину. Арифметичні операції кодуються в нуль-операндні інструкції. Стекова архітектура є невід'ємною частиною MISC-процесорів.

Реєстрова архітектура - характеризується вільним доступом до регістрів для вибірки всіх аргументів і записи результату. Елементарні арифметико-логічні операції в таких процесорах кодуються в дво-, або триоперандні інструкції (реєстр + регістр → регістр, іноді регістр результату збігається з джерелом одного з аргументів).

23.2 Структура мікропроцесора з акумулятором

Архітектура на базі акумулятора історично виникла однією з перших. У ній для зберігання одного з операндів арифметичної або логічної операції в процесорі є виділений регістр – акумулятор. В цей же регістр заноситься і результат операції. Оскільки адреса одного з операндів наперед визначена, в командах обробки досить явно вказати місце розташування тільки другого операнда. Спочатку обидва операнда зберігаються в основній пам'яті, і до виконання операції один з них потрібно завантажити в акумулятор. Після виконання команди результат заноситься в акумулятор, і якщо цей результат не є операндом для подальшої команди, то його потрібно зберегти в комірці пам'яті.

Типова архітектура на базі акумулятора показана на рис. 23.1.

Рисунок 23.1 –Архітектура мікропроцесора на базі акумулятора

Для завантаження в акумулятор вмісту комірки *x* передбачена команда завантаження *load x*. За цією командою інформація зчитується з комірки пам'яті *x*, вихід пам'яті підключається до входів акумулятора, і відбувається занесення лічених даних в акумулятор

Запис вмісту акумулятора в комірку *x* здійснюється командою збереження *store x*, при виконанні якої виходи акумулятора підключаються до шини, після чого інформація з шини записується в пам'ять. Таким чином один з операндів завжди є в акумуляторі.

Для виконання операції в АЛП проводиться зчитування одного з операндів з пам'яті в регістр даних. Другий операнд знаходиться в акумуляторі. виходи регістра даних і акумулятора підключаються до відповідних входів АЛП. Після закінчення запропонованої операції результат з виходу АЛП заноситься в акумулятор.

Перевагами акумуляторної архітектури є короткі команди і простота декодування команд. Однак наявність всього одного регістра породжує багаторазові звернення до основної пам'яті.

23.3 Структура МП з стеком

Основні вузли та інформаційні тракти одного з можливих варіантів на основі стекової архітектури показані на рис. 23.2.

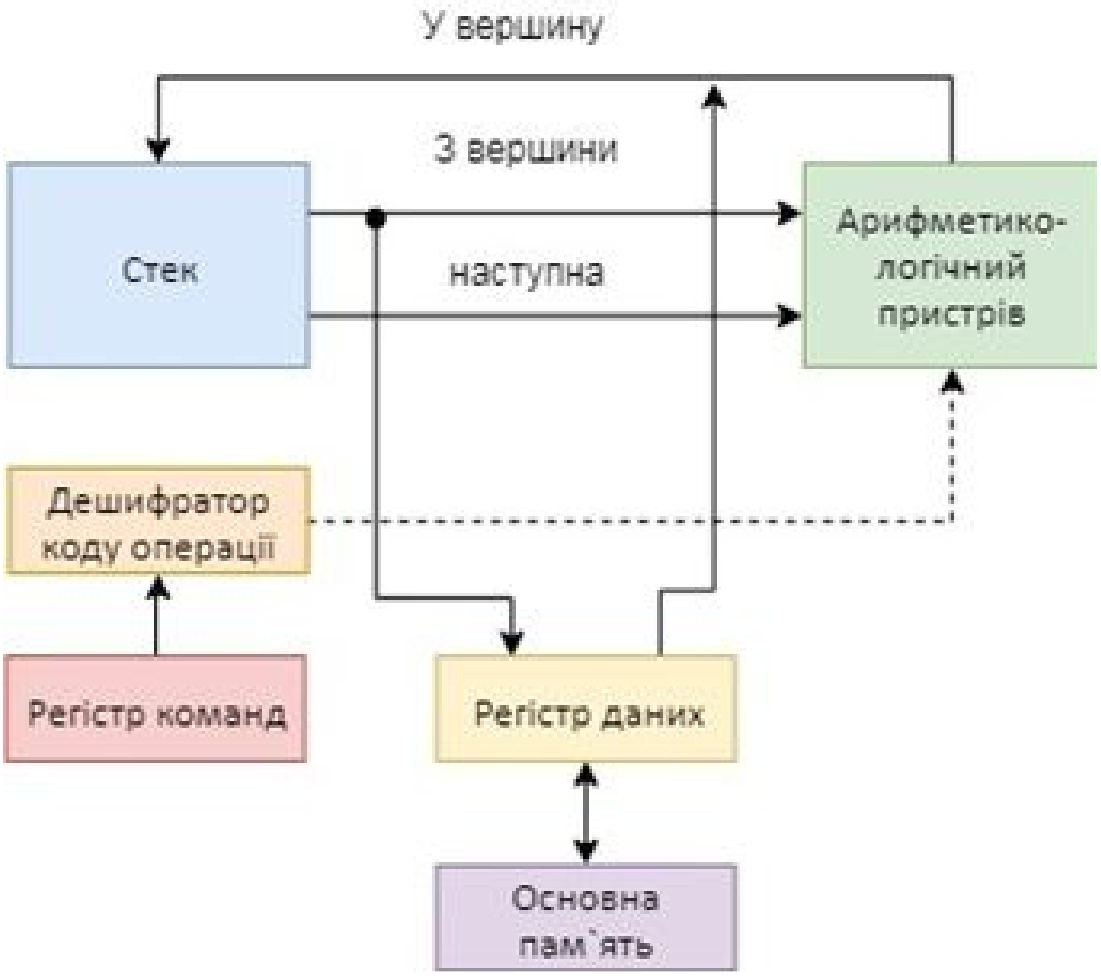

Рисунок 23.2 – Архітектура мікропроцесора на базі стека

Інформація може бути занесена в вершину стека з пам'яті або з АЛП. Для запису в стек вмісту комірки пам'яті з адресою *x* виконується команда *push x*, по якій інформація зчитується з комірки пам'яті, заноситься в регістр даних, а потім проштовхується в стек (рис. 23.3). Результат операції з АЛП заноситься в вершину стека автоматично.

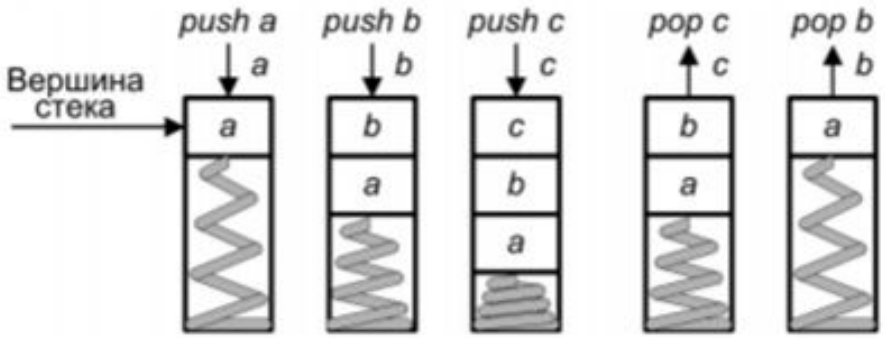

Рисунок 23.3 – Принцип дії стекової пам'яті

Збереження вмісту вершини стека в комірці пам'яті з адресою *x* проводиться командою *pop x*. За цією командою вміст верхньої комірки стека подається на шину, з якою і проводиться запис в осередок x, після чого весь вміст стека проштовхується на одну позицію вгору.

Для виконання арифметичної або логічної операції на вхід АЛП подається інформація, зчитана з двох верхніх комірки стека (при цьому вміст стека просувається на дві позиції вгору, тобто операнди зі стека витискуються). Результат операції заштовхується на вершину стека. Можливий варіант, коли результат відразу ж переписується в пам'ять за допомогою автоматично виконуваної операції *pop x*.

Верхні комірки стекової пам'яті, де зберігаються операнди і куди заноситься результат операції, як правило, роблять більш швидкодіючими і розміщують в процесорі, в той час як інша частина стека може розташовуватися в основний пам'яті і частково навіть на магнітному диску.

До переваг стекової архітектури слід віднести можливість скорочення адресної частини команд, оскільки всі операції проводяться через вершину стека, тобто адреси операнд і результату в командах арифметичної і логічної обробки інформації вказувати не потрібно. Код програми виходить компактним. Досить просто реалізується декодування команд. ї

З іншого боку, стекова архітектура за визначенням не передбачає довільного доступу до пам'яті, через що компілятору важко створити ефективний програмний код, хоча створення самих компіляторів спрощується. Крім того, стек стає «вузьким місцем» для підвищення продуктивності.

23.4 Реєстрова архітектура

В машинах даного типу процесор включає в себе масив регістрів загального призначення (РЗП). Розрядність регістрів зазвичай збігається з розміром машинного слова. До будь-якого регістру можна звернутися, вказавши його номер. Кількість РЗП в архітектурі типу CISC (англ. Complex Instruction Set Computer – архітектура з повним набором команд) зазвичай невелика (від 8 до 32), і для подання номера конкретного регістру необхідно не більше п'яти розрядів, завдяки чому в адресній частині команд обробки допускається одночасно вказати номери двох, а часто і трьох регістрів (двох регістрів операндів і регістра результату). Архітектура RISC (англ. Reduced Instruction Set Computer – архітектура зі скороченим набором команд) передбачає використання істотно більшого числа РЗП (до кількох сотень), однак типова для таких процесорів довжина команди (зазвичай 32 розряду) дозволяє визначити в команді до трьох регістрів. Можливу структуру та інформаційні тракти з реєстрової архітектурою системи команд ілюструє рис. 23.4.

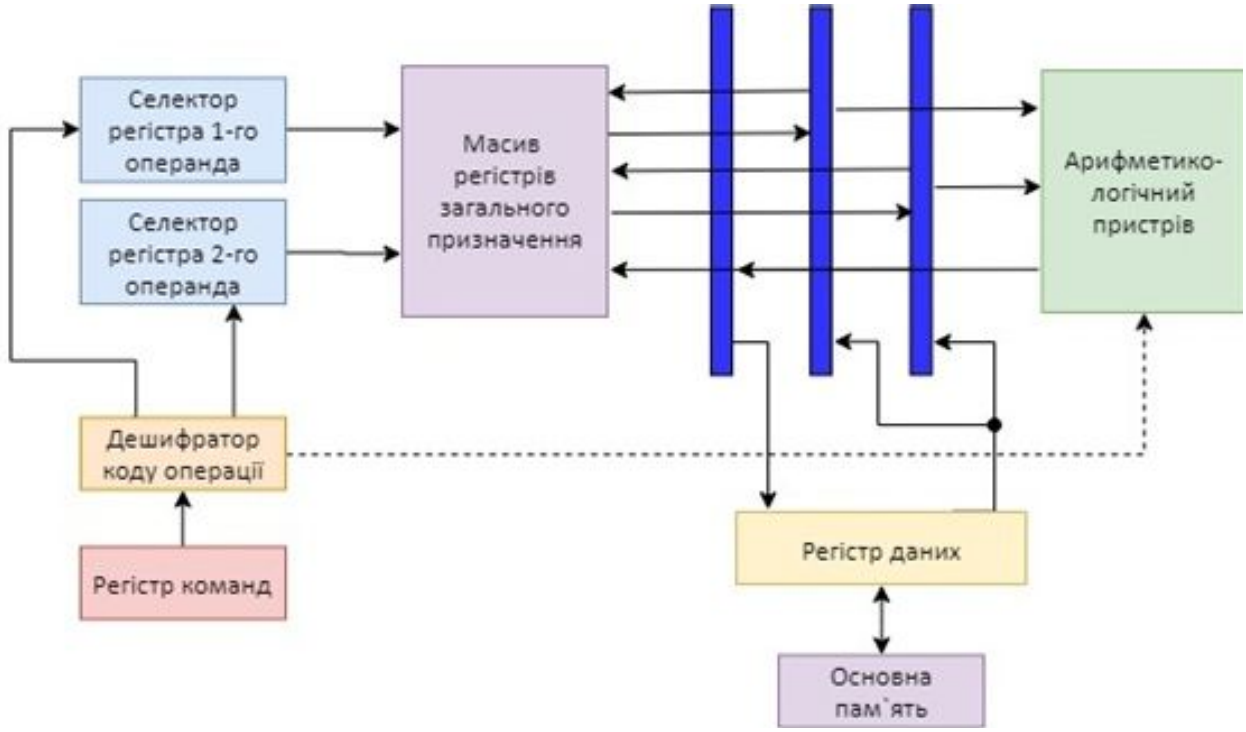

Рисунок 23.4 –Архітектура на базі регістрів загального призначення

Операції завантаження регістрів з пам'яті і збереження вмісту регістрів в пам'яті ідентичні таким же операціям з акумулятором. Відмінність полягає в етапі вибору потрібного регістра, що забезпечується відповідними селекторами.

Виконання операції в АЛП включає в себе:

- визначення місця розташування першого операнда (регістр або пам'ять);
- вибір регістра першого операнда або зчитування першого операнда з пам'яті;
- визначення місця розташування другого операнда (регістр або пам'ять);
- вибір регістра другого операнда або зчитування другого операнда з пам'яті;
- подачу на вхід АЛП операндів і виконання операції;
- визначення місця розташування результату (реєстр або пам'ять);
- вибір регістра результату або комірки пам'яті і занесення результату операції з АЛП.

Звернемо увагу на те, що між АЛП і реєстрових файлом повинні бути три шини. Дві з трьох шин, розташованих між масивом РЗП і АЛП, забезпечують передачу в арифметико-логічний пристрій операндів, що зберігаються в регістрах загального призначення або осередках пам'яті. Третя служить для занесення результату в виділений для цього регістр або елемент пам'яті. Ці ж шини дозволяють завантажити в регістри комірки основної пам'яті і зберегти в ОП інформацію, що знаходиться в РЗП.

До переваг реєстрової архітектури мікропроцесора слід віднести: компактність отримуваного коду, високу швидкість обчислень за рахунок заміни звернень до основної пам'яті на звернення до швидких регістрів. З іншого боку, дана архітектура вимагає довших команд в порівнянні з акумуляторної архітектурою.

У наші дні цей вид архітектури системи команд є переважаючим, зокрема таку архітектуру мають сучасні персональні комп'ютери

23.5 Архітектура з виділеним доступом до пам'яті

В архітектурі з виділеним доступом до пам'яті звернення до основної пам'яті можливе тільки за допомогою двох спеціальних команд: *load* і *store*. В англійській транскрипції дану архітектуру називають *Load/Store architecture*. Команда *load* (завантаження) забезпечує зчитування значення з основної пам'яті і занесення його в регістр процесора (в команді зазвичай вказується адреса комірки пам'яті і номер регістра). Пересилання інформації в протилежному напрямку проводиться командою *store* (збереження). Операнди у всіх командах обробки інформації можуть перебувати тільки в регістрах процесора (найчастіше в регістрах загального призначення). Результат операції також заноситься в регістр.

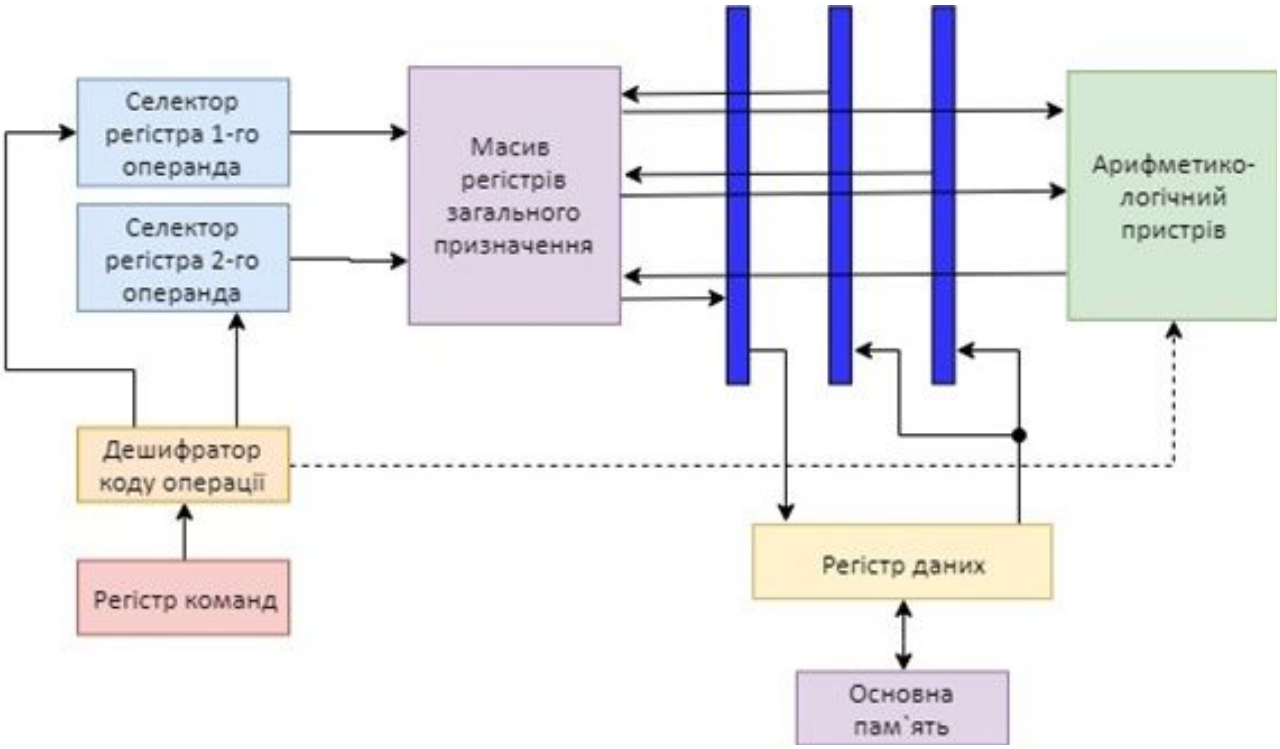

Рисунок 23.5 – Архітектура з виділеним доступом до пам'яті