Лекція

з дисципліни: **«Комп'ютерна схемотехніка та архітектура комп'ютерів»**

на тему: «Однотактний процесор»

ПЛАН ЛЕКЦІЇ

1. Архітектурний стан.
2. Однотактний процесор.
3. Однотактний пристрій управління

ЛІТЕРАТУРА

1. Рябенький В. М. Цифрова схемотехніка / Рябенький В. М., Жуйков В. Я., Гулий В. Д. – Львів: "Новий Світ-2000", 2020. — 736 с.
2. Девід М. Харріс і Сара Л. Харріс. Цифрова схемотехніка та архітектура комп'ютера / пер. з англ. 2-е вид. - Morgan Kaufman, 2013. - 1621 с.
3. Матвієнко М. П. Архітектура комп'ютера. Навчальний посібник. / Матвієнко М. П., Розен В. П., Закладний О. М.— К: Видавництво Ліра-К, 2016. — 264 с.

1.Архітектурний стан

Комп'ютерна архітектура визначається набором команд і *архітектурним станом. Архітектурний стан* процесора MIPS визначається вмістом лічильника команд (program counter, PC) і 32 видимих програмісту регістрів, тому будь-який процесор, який реалізує архітектуру MIPS, незалежно від його мікроархітектури зобов'язаний мати лічильник команд і рівно 32 регістра. Знаючи поточний архітектурний стан, процесор точно знає, яку операцію над якими даними треба виконати для отримання нового архітектурного стану.

Щоб мікроархітектура залишалася простою для розуміння, ми розглянемо тільки невелику підмножину набору команд MIPS, а саме:

1. Арифметичні і логічні команди типу R: add, sub, and, or, slt
2. Команди доступу в пам'ять: lw, sw
3. Команди умовного переходу: beq

Мікроархітектуру можна розбити на дві взаємодіючих частини: тракт даних і пристрій управління. Тракт даних працює зі словами даних. Він містить такі блоки, як пам'ять, регістри, АЛП та мультиплексори.

Пристрій управління отримує поточну команду з тракту даних і у відповідь каже йому, як саме виконувати цю команду. Зокрема, пристрій управління генерує адресні сигнали для мультиплексорів, сигнали дозволу роботи для регістрів і сигнали дозволу запису в пам'ять.

На Рис.1 показані чотири елементи стану: лічильник команд, регістровий файл, пам'ять команд і пам'ять даних. Ці елементи включають пам'ять для зберігання команд і даних і блоки для зберігання архітектурного стану, тобто лічильник команд і видимі програмісту регістри.

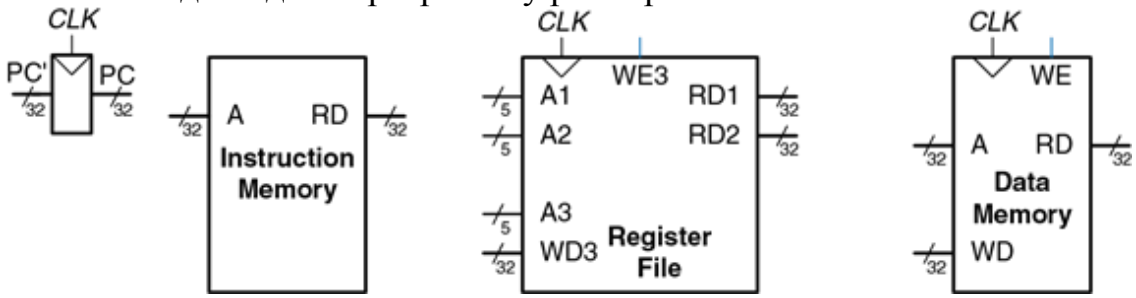


Рис.1

У елементів, що зберігають стан системи, зазвичай є сигнал скидання, який встановлює їх в відомий стан в момент вмикання.

Лічильник команд - це звичайний 32-бітний регістр. Його вихід (PC) містить адресу поточної команди. Його вхід (PC') містить адресу наступної команди.

Пам'ять команд має один порт для зчитування. На адресний вхід A подається 32-бітову адресу команди, після чого на виході RD з'являється 32-бітове число (тобто команда), прочитане з пам'яті за цією адресою.

Регістровий файл, який містить 32 елемента по 32 біта кожен, має два порти зчитування і один порт запису даних. Порти зчитування мають п’ятибітні входи адреси A1 і A2, кожен з яких визначає один з 32 регістрів як джерело даних для команди. Кожен з портів читає 32-бітове значення з регістра і подає його на виходи RD1 і RD2 відповідно. Порт запису отримує п’ятибітну адресу

регістра на адресний вхід A3, 32-бітове число на вхід даних WD, сигнал дозволу запису WE3 і тактовий сигнал. Якщо сигнал дозволу запису дорівнює одиниці, то регістровий файл записує дані в зазначений регістр по позитивному фронту тактового сигналу.

Пам'ять даних має один порт читання / запису. Якщо сигнал дозволу запису WE дорівнює одиниці, то дані з входу WD записуються в комірку пам'яті з адресою A по позитивному фронту тактового сигналу. Якщо ж сигнал дозволу запису дорівнює нулю, то дані з комірки з адресою A подаються на вихід RD.

Читання з пам'яті команд, регістрового файлу і пам'яті даних відбувається асинхронно, тобто незалежно від тактового сигналу. Іншими словами, відразу ж після зміни значення на адресному вході на виході RD з'являються нові дані. Це відбувається не миттєво, так як існує затримка поширення сигналу, проте тактовий сигнал для читання не потрібний. Запис же проводиться виключно по позитивному фронту тактового сигналу.

2. Однотактний процесор

Однотактна мікроархітектура виконує всю команду за один такт. Її принцип роботи легко пояснити, а пристрій управління досить простий. Однак тривалість такту обмежена найповільнішої командою.

2.1.Однотактний тракт даних

Лічильник команд (Program Counter, PC) містить адресу команди, яку треба виконати. Першим кроком треба прочитати цю команду з пам'яті команд. Як показано на Рис.2, лічильник команд безпосередньо підключений до адресного входу пам'яті команд. Команда, прочитана, або обрана (fetched), з пам'яті команд - це 32-бітна команда, зазначена на малюнку як Instr.

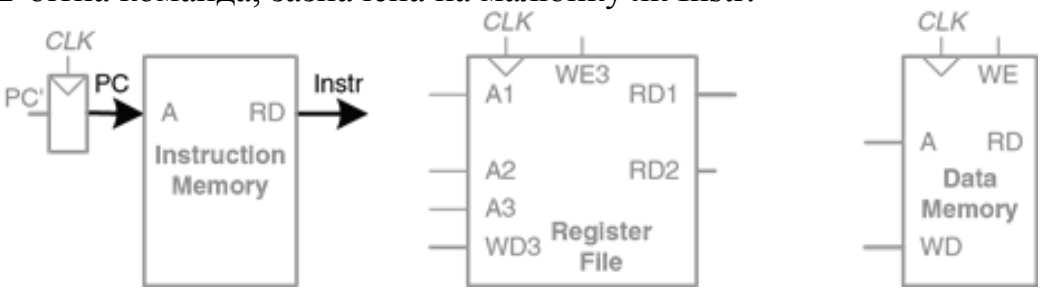


Рис.2 Вибір команди з пам’яті

Подальші дії процесора будуть залежати від того, яка саме команда була обрана.

Для команди lw наступним кроком буде прочитано регістр операнда (source register), що містить так звану базову адресу. Номер цього регістра вказано в полі rs (Instr25: 21). Ці п'ять бітів підключені до адресного входу першого порту (A1) регістрового файлу, як показано на Рис. 3. Значення, прочитане з регістрового файлу, з'являється на його виході RD1. Команді lw також потрібно зміщення (offset) - число, яке буде додано до базової адреси. Зсув передається як безпосередній операнд (immediate), тобто знаходиться безпосередньо в полі Instr15: 0, займаючи молодші 16 біт. Так як 16-бітове число може бути як позитивним, так і негативним, то над ним повинна бути виконана

операція знакового розширення до 32 біт, як показано на Рис. 4. Отримане 32-бітове розширене значення називається SignImm.

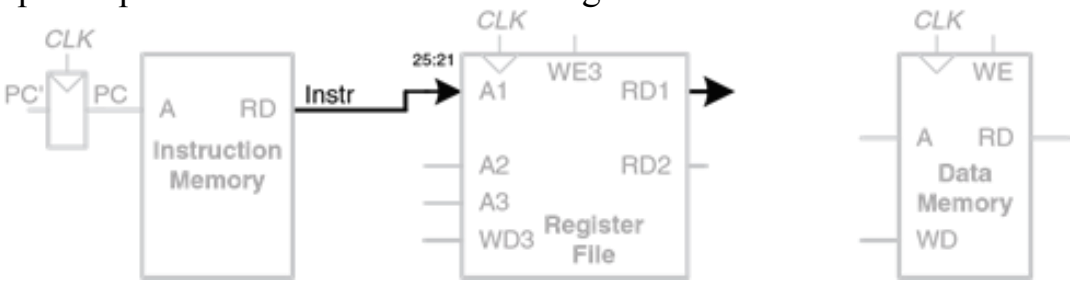


Рис. 3

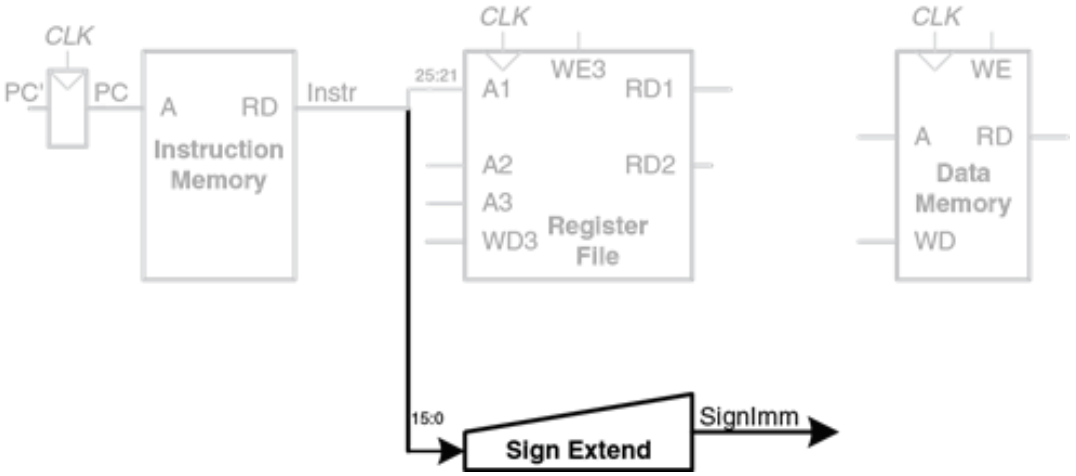


Рис. 4.

Знакове розширення полягає в тому, що знаковий біт (він же старший біт) числа, яке розширюється, просто копіюється в усі старші біти розширеного числа.

Процесор повинен додати зміщення до базової адреси, щоб отримати адресу, за якою буде виконано читання з пам'яті. Для підсумовування ми додаємо в тракт даних АЛП (ALU), як показано на Рис. 5. АЛП отримує на входи два операнда, SrcA і SrcB.

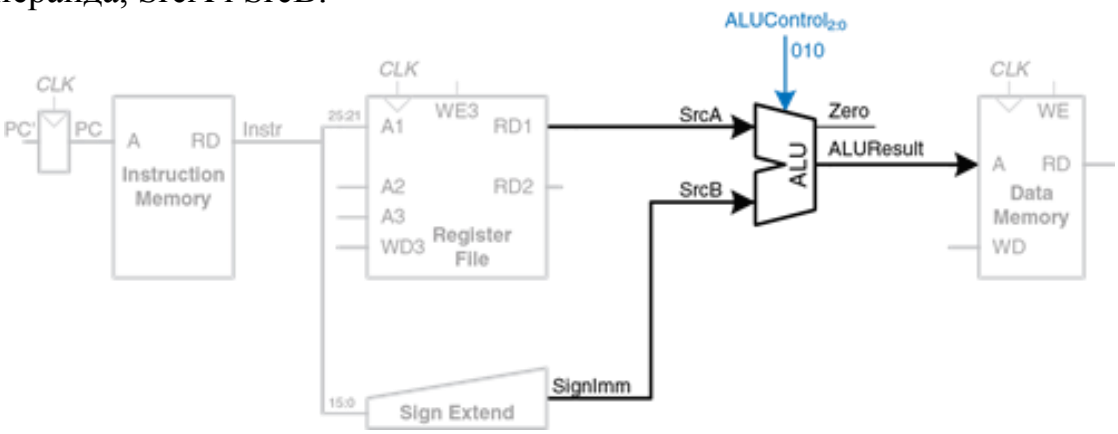
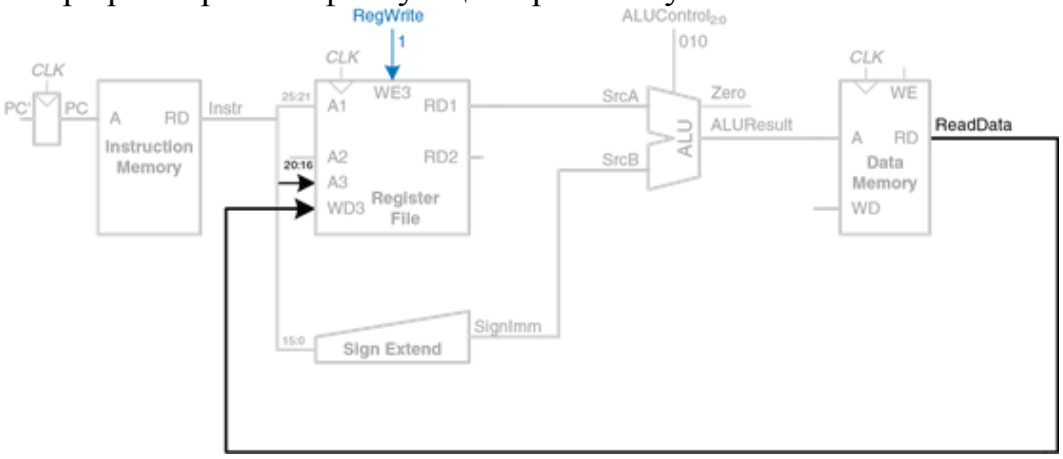


Рис. 5. Визначення адреси даних в пам’яті.

SrcA подається прямо з регістрового файлу, а SrcB - зі зміщення зі знаковим розширенням. Трьохбітний керуючий сигнал ALUControl вказує АЛП, яку операцію треба виконати. На виході з АЛУ виходить 32-бітний результат ALUResult і прапор нуля (Zero flag), який встановлюється в одиницю, якщо ALUResult дорівнює нулю. Для команди lw сигнал ALUControl має дорівнювати

010 - в цьому випадку зміщення буде додано до базової адреси. Далі ALUResult подається на адресний вхід пам'яті даних, як показано на Рис. 5.

Значення, прочитане з пам'яті даних, потрапляє на шину ReadData, після чого записується назад в регістровий файл в кінці такту, як показано на Рис.6. Третій порт регістрового файлу - це порт запису.



Регістр результату (destination register), в який команда lw запише прочитане з пам'яті значення, визначається полем rt (Instr20: 16), що підключене до адресного входу третього порту (A3) регістрового файлу. Шина ReadData підключена до входу даних третього порту (WD3). Керуючий сигнал RegWrite, в свою чергу, з'єднаний зі входом дозволу запису третього порту (WE3) і активний під час виконання команди lw, щоб прочитане значення було записано в регістровий файл.

Одночасно з виконанням команди процесор повинен обчислити адресу наступної команди PC '. Так як команди 32-бітові, тобто чотирибайтові, то адреса наступної команди дорівнює PC + 4. На Рис. 7 показано, що для цього нам потрібен ще один суматор. Нова адреса записується в лічильник команд в момент приходу позитивного фронту тактового сигналу. На цьому створення тракту даних завершено.

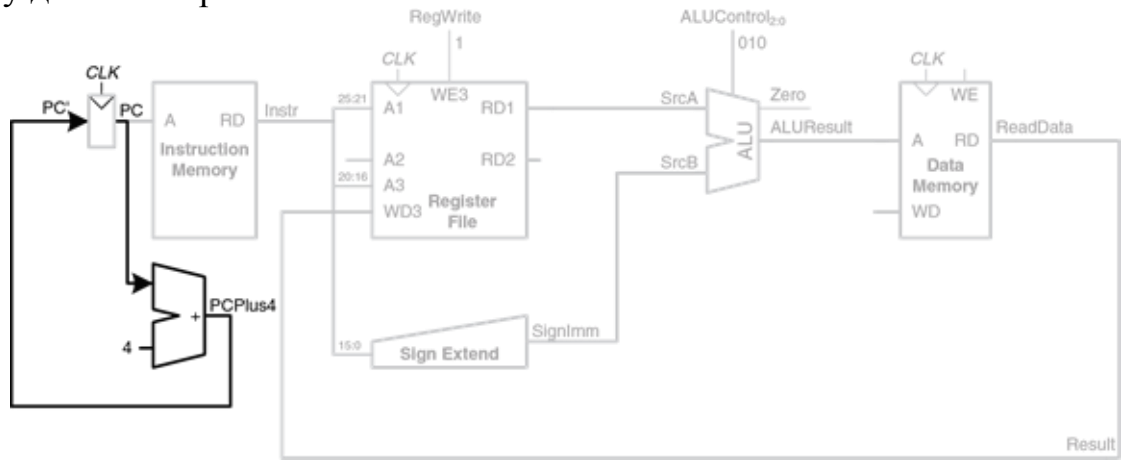


Рис. 7

Тракт даних виконання команди sw.

Як і команда lw, sw читає базову адресу з першого порту регістрового файлу і знаково розширює зміщення, яке передається як безпосередній операнд. АЛП складає базовий адрес зі зміщенням, щоб отримати адресу в пам'яті. Всі ці функції вже реалізовані в тракті даних.

Команда sw, на відміну від lw, читає з регістрового файлу ще один регістр і записує його вміст у пам'ять даних, як показано на Рис. 8. Номер регістра вказується в поле rt (Instr20: 16). Ці п'ять біт підключені до другого порту (A2) регістрового файлу. Прочитане значення з'являється на виході RD2 і потрапляє на вхід запису в пам'ять даних. Вхід дозволу запису (WE) управляється сигналом MemWrite. Для команди sw сигнал MemWrite = 1, щоб дані були записані в пам'ять; ALUControl = 010, щоб базова адреса була просумована зі зміщенням; і RegWrite = 0, тому що команда нічого не пише в регістровий файл. Зауважте, що ReadData читається з пам'яті в будь-якому випадку, але прочитане значення ігнорується, так як RegWrite = 0.

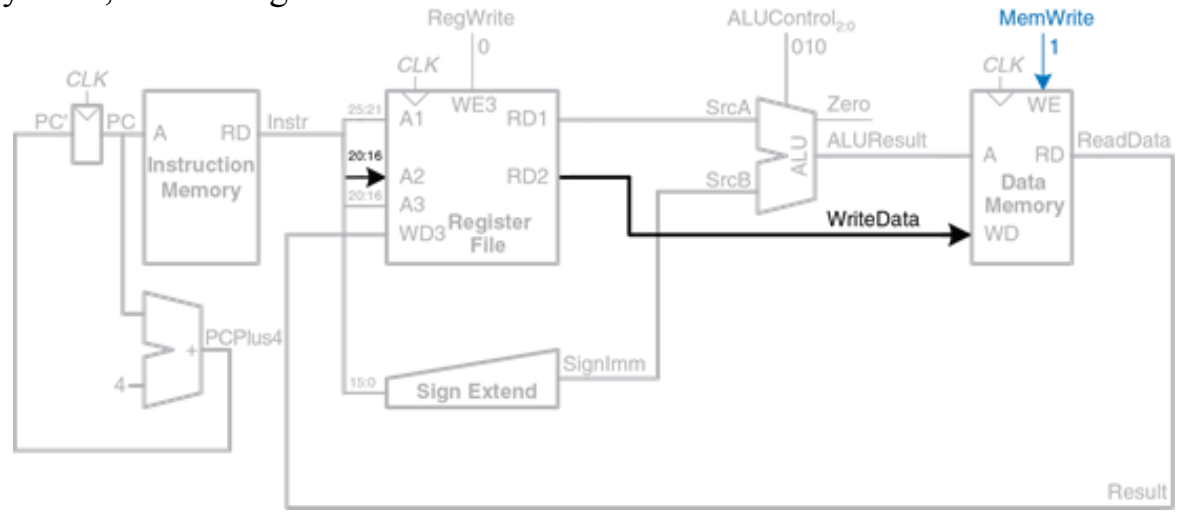


Рис. 8. Запис командою sw даних в пам’ять

Команди типу R - add, sub, and, or, і slt.

Всі ці команди читають два регістра з регістрового файлу, виконують над ними якісь операції в АЛП і записують результат назад в третій регістр. Єдина відмінність цих команд - в типі операції. Таким чином, всі вони можуть бути виконані однією і тією ж апаратурою, використовуючи тільки лише різні значення керуючого сигналу ALUControl.

Оновлений тракт даних з підтримкою команд типу R показаний на Рис. 9. Вміст двох регістрів читається з регістрового файлу і подається на входи АЛП. На Рис. 8 операнд SrcB завжди дорівнював безпосередньому значенням SignImm з розширенням знаку. Тепер необхідно додати мультиплексор, щоб була можливість подати на вхід АЛП або SignImm, або вихід RD2 регістрового файлу. Мультиплексор управляється новим сигналом ALUSrc. ALUSrc дорівнює нулю для команд типу R - в цьому випадку на вхід АЛП подається значення з регістрового файлу. Для команд lw і sw ALUSrc дорівнює одиниці, і на вхід АЛП подається SignImm. Додавання мультиплексорів в тракт даних - дуже зручний спосіб вибрати один з декількох джерел даних і подати його на вхід якого-небудь блоку процесора.

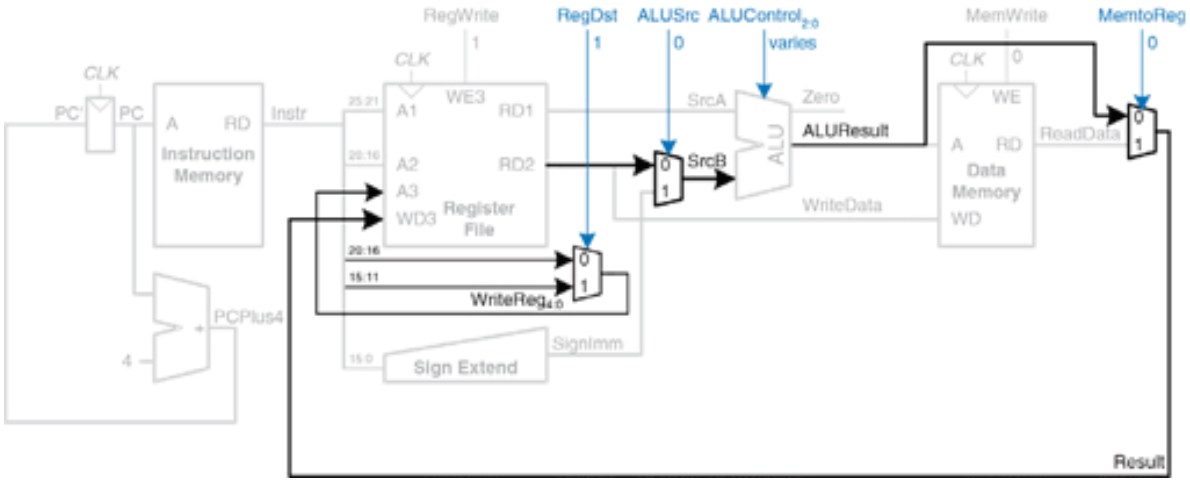


Рис. 9

На Рис. 8 порт запису регістрового файлу був завжди підключений до пам'яті даних. Однак команди типу R повинні записувати в регістровий файл значення ALUResult. Щоб вибирати між ReadData і ALUResult, ми додаємо ще один мультиплексор, вихід якого будемо називати Result. Цей мультиплексор управляється ще одним новим сигналом - MemtoReg. MemtoReg дорівнює нулю для команд типу R - в цьому випадку Result приймає значення ALUResult. Для команди lw MemtoReg дорівнює одиниці, а Result приймає значення ReadData. Для команди sw значення MemtoReg не має ніякого значення, так як sw нічого в регістровий файл не пише.

Аналогічно, на Рис. 8 номер регістра в регістровому файлі, куди потрібно було записати дані, визначався полем rt (Instr20: 16). Для команд типу R, однак, номер регістра задається полем rd (Instr15: 11), через це ми додаємо третій мультиплексор, щоб присвоювати сигналу WriteReg значення з потрібного поля. Цей мультиплексор управляється сигналом RegDst. RegDst дорівнює одиниці для команд типу R - в цьому випадку WriteReg приймає значення поля rd (Instr15: 11). Для команди lw RegDst дорівнює нулю, а WriteReg приймає значення поля rt (Instr20: 16). Для команди sw значення RegDst не має ніякого значення, так як sw нічого в регістровий файл не пише.

Виконання команди beq.

Ця команда порівнює два регістра, і, якщо вони рівні, то додає зсув до лічильника команд PC, виконуючи, таким чином, умовний перехід. Нагадаємо, що зсув - це позитивне чи негативне число, яке передається як безпосередній операнд в поле Instr15: 0.

Зсув вказує, скільки команд потрібно перестрибнути, перш ніж продовжити виконання програми. Таким чином, над значенням безпосереднього операнда треба виконати операцію знакового розширення, після чого помножити його на чотири, щоб отримати нове значення лічильника команд: PC '= PC + 4 + SignImm × 4

Зміни, які потрібно внести в тракт даних, показані на Рис. 10. Нове значення лічильника команд у випадку виконаного умовного переходу (PCBranch) обчислюється шляхом зсуву SignImm вліво на два розряди і подальшого складання з PCPlus4. Зсув вліво на два розряди - це легкий спосіб

множення на чотири, так як зсув на константне число розрядів не вимагає ніяких логічних елементів. Два регістра порівнюються шляхом віднімання одного з іншого в АЛП. Якщо ALUResult дорівнює нулю, про що сигналізує прапор нуля, то регістри рівні. Потрібно додати мультиплексор, щоб вибрати, який саме значення привласнити PC'- PCPlus4 або PCBranch. PCBranch використовується тоді, коли виконується команда умовного переходу і встановлений прапор нуля. Таким чином, сигнал Branch дорівнює одиниці для команди beq і нулю для інших команд. Для beq сигнали ALUControl = 110, що означає, що АЛП має віднімати. ALUSrc = 0, щоб операнд SrcB був прочитаний з регістрового файлу. RegWrite і MemWrite дорівнюють нулю, так як команда умовного переходу нічого не пише ні в регістровий файл, ні в пам'ять. Значення RegDst і MemtoReg нас не цікавлять, тому що в регістровий файл нічого не пишеться.

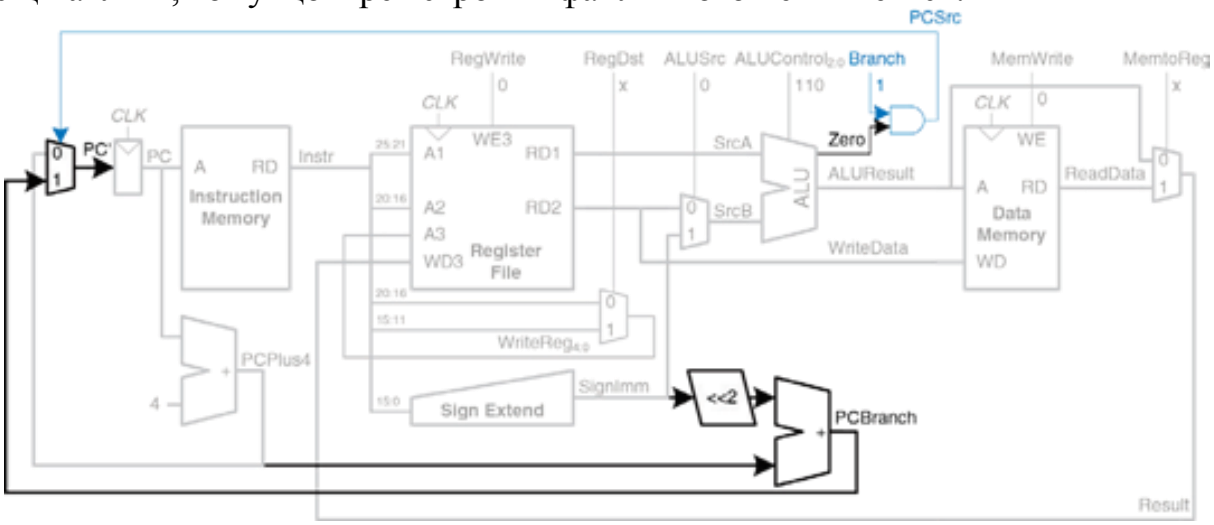


Рис. 10

3. Однотактний пристрій управління

Пристрій управління формує керуючі сигнали на основі полів opcode і funct, присутніх в кожній команді (Instr31: 26 і Instr5: 0 відповідно). На Рис. 11 показаний однотактний процесор MIPS з пристроєм управління, підключеним до тракту даних.

Основна частина інформації для пристрою управління береться з поля opcode, але команди типу R також використовують і поле funct для визначення операції в АЛП. Таким чином, для спрощення розробки ми поділимо пристрій управління на дві частини, що містять комбінаційну логіку, як показано на Рис. 12.

Основний дешифратор обчислює значення більшості виходів на основі поля opcode. Він також формує двухбітний сигнал ALUOp. Дешифратор АЛП (ALU decoder) використовує ALUOp спільно з полем funct для визначення стану ALUControl.

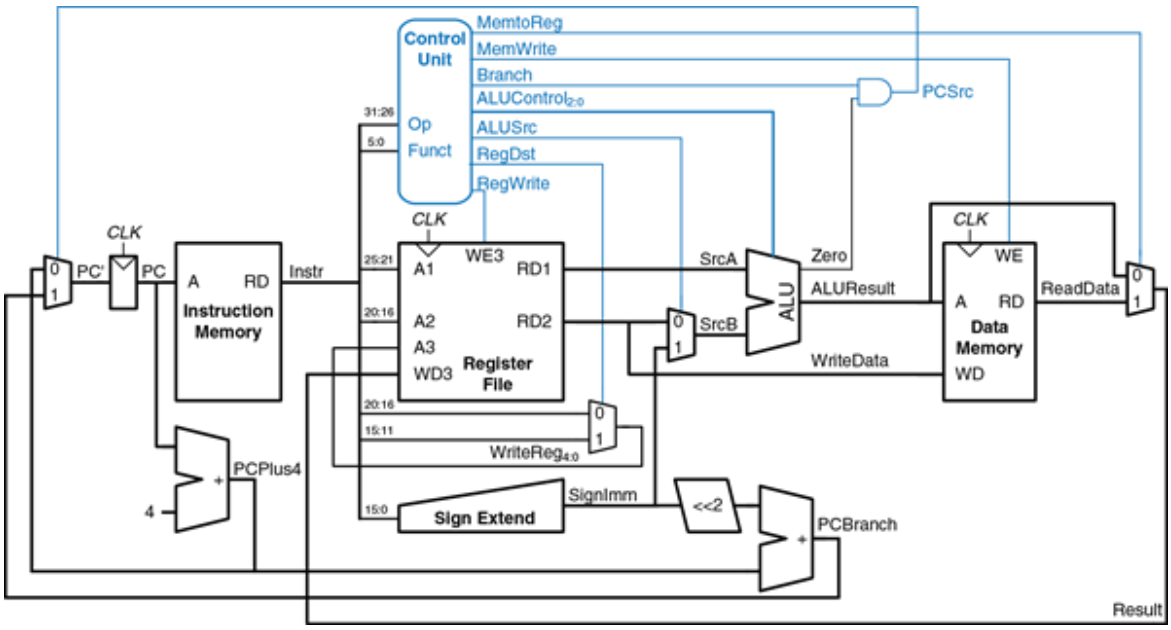


Рис. 11

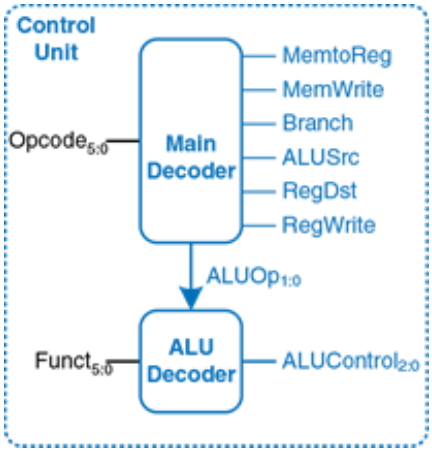


Рис.12

Коли сигнал ALUOp дорівнює 00 або 01, АЛП має додавати або віднімати відповідно. Коли він дорівнює 10, то значення ALUControl має визначатися полем funct. Для команд типу R, які ми додали, перші два біта поля funct завжди рівні 10