Львівський державний університет безпеки життєдіяльності

Кафедра інформаційних технологій та
систем електронних комунікацій

Лекція

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

на тему: «Багатоактний процесор»

Львів

МЕТА ЛЕКЦІЇ

Навчальна мета: Ознайомити курсантів(студентів) з будовою процесора. Розглянути однотактні та багатоактні пристрої керування.

Виховна мета: Виховувати прагнення оволодівати знаннями та відчуття перед суспільством за отримані знання

Розвиткова мета: Розвиток вміння використання процесорів для вирішення технічних завдань.

Забезпечення заняття

1.Роздатковий матеріал
2.Технічні засоби навчання: мультимедійний проектор.
3.Місце проведення заняття: лабораторія телекомунікаційних систем.

ПЛАН ЛЕКЦІЇ

1. Багатотактний процесор
2. Багатотактний 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. Багатотактний процесор

У однотактного процесора три основних проблеми. По-перше, період його тактового сигналу повинен бути досить великим, щоб встигла виконатися найповільніша команда (lw), незважаючи на те, що більшість інших команд набагато швидші. По-друге, йому потрібно три суматори (один для АЛП і два для обчислення нового значення лічильника команд); суматори, особливо швидкі, вимагають безлічі логічних елементів, що робить їх відносно дорогими схемами. По-третє, йому потрібна окрема пам'ять команд і даних, що часто нереально. У більшості комп'ютерів використовують загальну пам'ять для команд і даних, доступну для читання і запису.

Один із способів вирішити ці проблеми - використовувати багатотактний процесор, в якому виконання кожної команди розбивається на кілька етапів. На кожному етапі процесор може читати або писати дані в пам'ять або регістровий файл, або виконувати будь-яку операцію в АЛП. У різних команд в цьому випадку буде різна кількість етапів, так що прості команди зможуть виконуватися швидше, ніж складні. Знадобиться лише один суматор; на різних етапах він може використовуватися для різних цілей. Також процесор зможе обходитися спільною пам'яттю для команд і даних. Команди будуть вибиратися на першому етапі, а читання або запис даних будуть відбуватися на одному з наступних етапів.

Багатотактний тракт даних

На Рис. 13 зображені елементи, що зберігають стан - пам'ять і архітектурний стан процесора. У однотактному процесорі ми використовували роздільну пам'ять команд і даних, тому що потрібно було за один і той же такт і читати з пам'яті команд, і звертатися до пам'яті даних. Тепер ми будемо використовувати загальну пам'ять, що зберігає і команди, і дані.

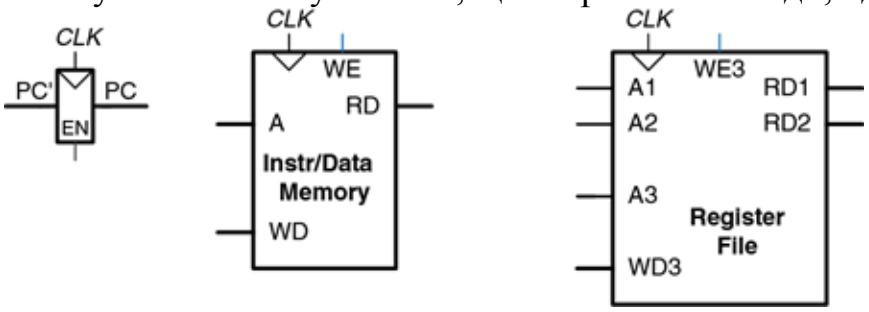

Рис. 13

Як показано на Рис. 14, лічильник команд безпосередньо приєднаний до адресного входу пам'яті команд. Прочитана з пам'яті команда зберігається в тимчасовий (неархітектурний) регістр команд (Instruction Register), так що ми зможемо використовувати її в наступних тактах. Сигнал дозволу запису в регістр команд назвемо IRWrite і будемо використовувати його, коли буде потрібно оновити команду, яка знаходиться в регістрі.

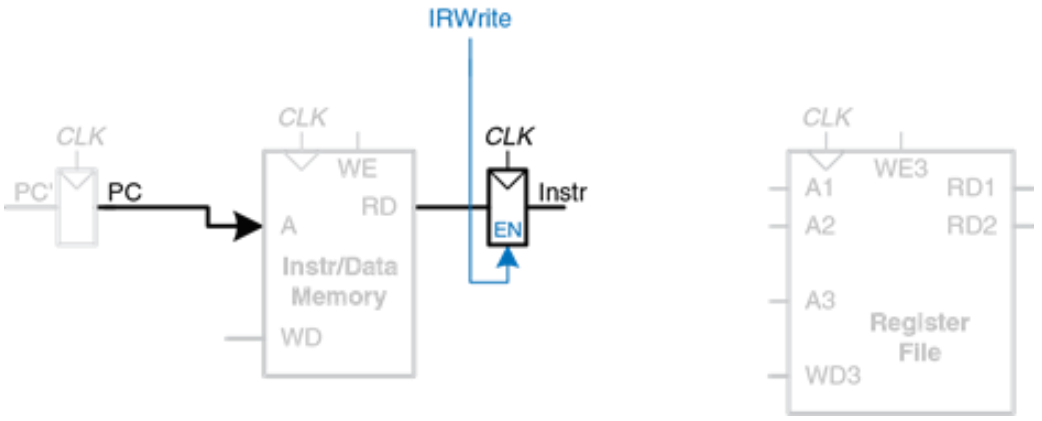

Рис. 14

Для команди lw наступним кроком буде читання регістра, що містить базову адресу. Номер регістра вказується в полі rs (Instr25: 21) і подається на адресний вхід першого порту (A1) регістрового файлу, як показано на Рис. 15. Значення, прочитане з регістрового файлу, з'являється на його виході RD1, після чого зберігається в тимчасовий регістр A.

Команді lw також потрібний зсув, який представляє собою безпосередній операнд (immediate), що знаходиться в одному з полів команди, Instr15: 0. Над цим операндом виконується операція знакового розширення, в результаті чого виходить 32-бітове число SignImm

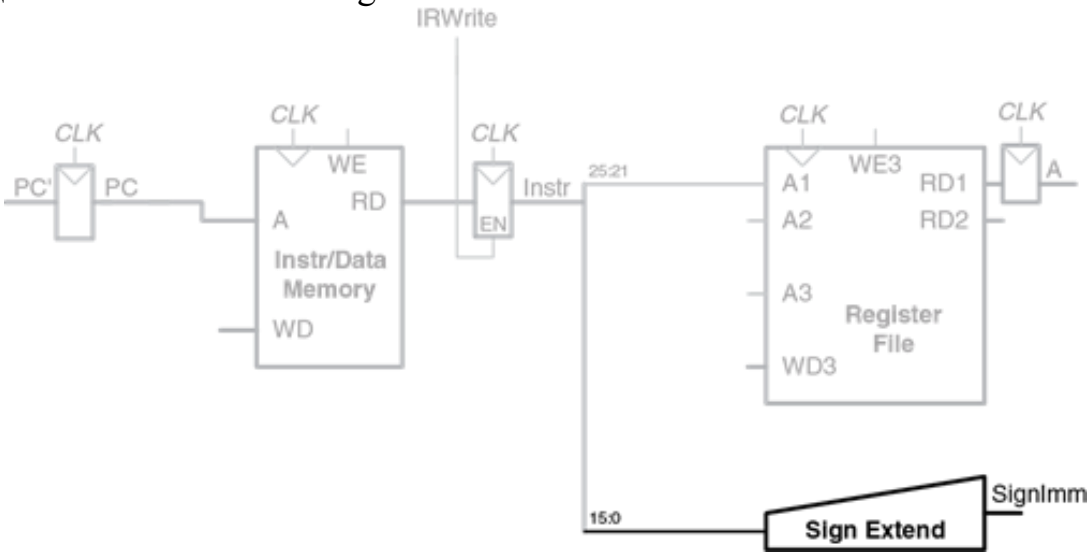

Рис. 15

Адресу, за якою ми повинні читати з пам'яті, отримуємо шляхом додавання базової адреси і зсуву. Для додавання ми використовуємо АЛП, як показано на Рис. 16. Щоб АЛП виконало додавання, керуючий сигнал ALUControl має дорівнювати 010. ALUResult зберігається у тимчасовому регістрі ALUOut.

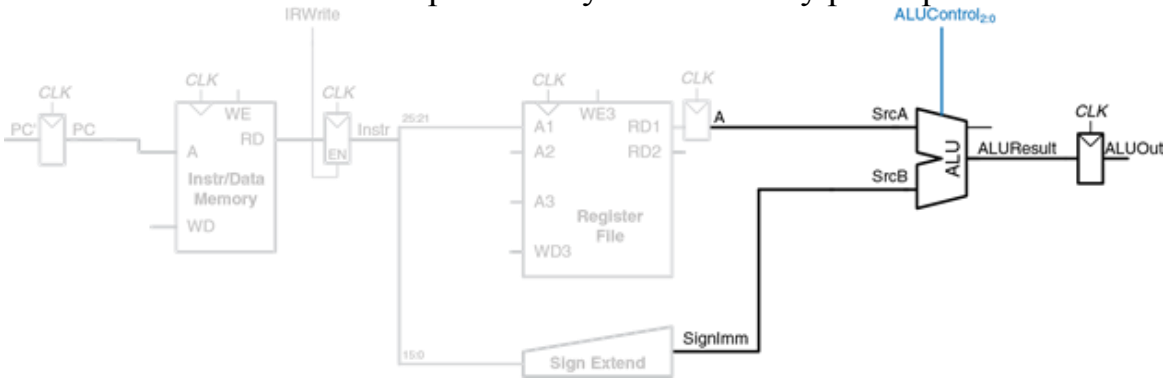

Рис. 16

Наступним кроком ми повинні прочитати дані з пам'яті, використовуючи тільки що отриману адресу. Для цього перед адресним входом пам'яті необхідно додати мультиплексор, щоб в якості адреси Adr можна було використовувати або PC, або ALUOut, як показано на Рис. 17.

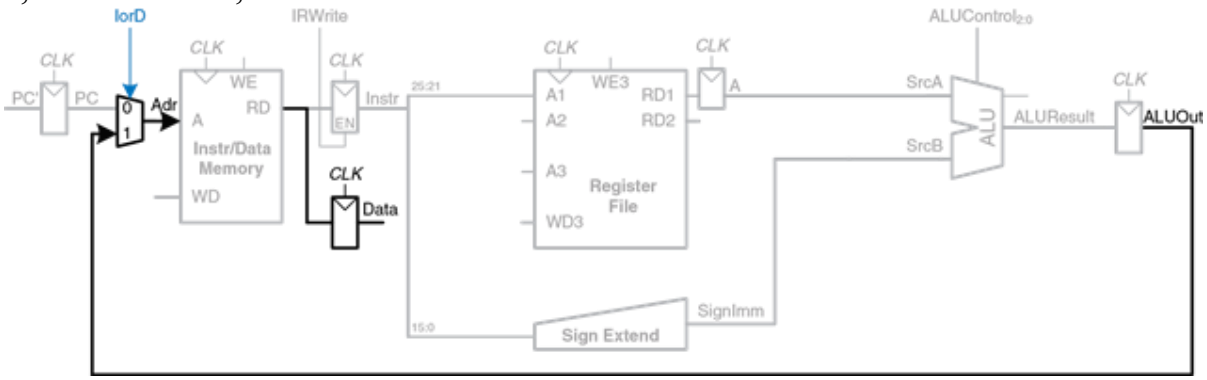

Рис. 17

Цей мультиплексор керується сигналом IorD, що показує, чи повинна бути подана адреса команди або адреса даних. Прочитані з пам'яті дані зберігаються в тимчасовому регістрі Data. Зауважте, що мультиплексор адреси дозволяє нам повторно використовувати пам'ять під час виконання команди lw. Спочатку в якості адреси ми використовуємо PC, що дозволяє вибрати команду. Потім в якості адреси ми використовуємо ALUOut і читаємо дані. Таким чином, керуючий сигнал IorD повинен приймати різні значення на різних етапах виконання команди.

Дані повинні бути записані в регістровий файл, як показано на Рис. 18. Номер регістра результату визначається полем rt (Instr20: 16).

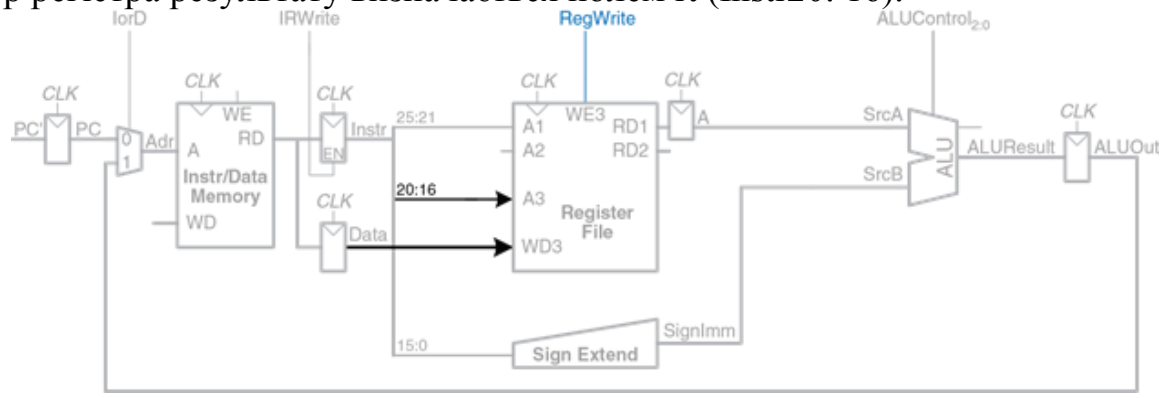

Рис.18

За той час, поки виконуються всі перераховані вище операції, процесор повинен збільшити лічильник команд на чотири. У однотактному процесорі для цього нам потрібен був окремий суматор. У багатотактному процесорі ми можемо використовувати вже наявний АЛП на одному з перших етапів, поки він не використовується. Для цього знадобиться додати пару мультиплексорів, які дозволять подавати на входи АЛП вміст лічильника команд PC і константу 4, як показано на Рис. 19.

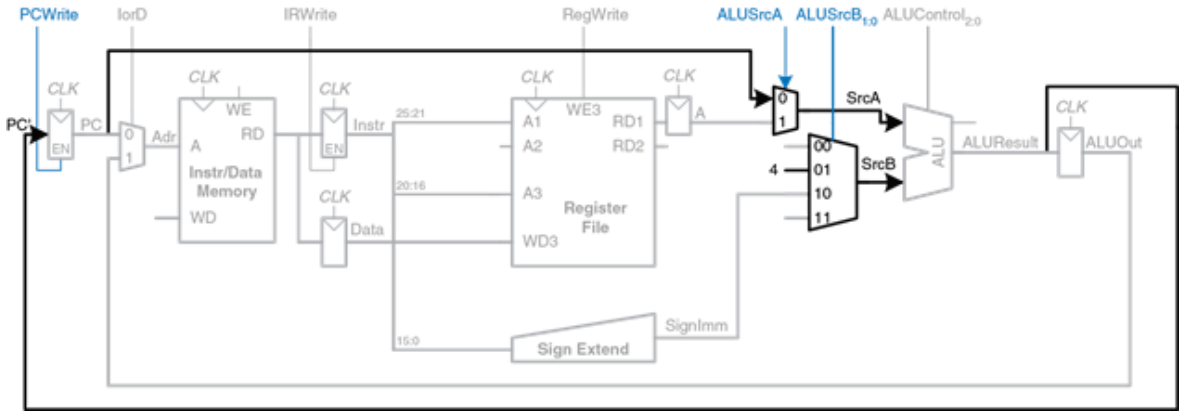

Рис. 19

Двухвходовий мультиплексор (мультиплексор 2: 1), керований сигналом ALUSrcA, подає на SrcA або PC, або регістр A. Чотиривходовий мультиплексор (мультиплексор 4: 1), керований сигналом ALUSrcB, подає на SrcB або константу 4, або SignImm. Для того, щоб оновити лічильник команд, АЛП сумує SrcA (PC) і SrcB (4) і записує отриманий результат в лічильник команд. Керуючий сигнал PCWrite дозволяє запис в лічильник команд тільки на тих тактах, де це необхідно.

Команда sw.

Як і команда lw, sw читає базову адресу з першого порту регістрового файлу і виконує знакове розширення безпосереднього операнда, після чого АЛУ сумує їх, отримуючи адресу для запису в пам'ять. Всі ці функції вже є в тракті даних.

Єдина відмінність sw - це те, що ми повинні прочитати ще один регістр з регістрового файлу і записати його вміст в пам'ять, як показано на Рис. 20. Номер регістра вказано в полі rt (Instr20: 16), що підключений до другого порту регістрового файлу. Прочитане значення зберігається у тимчасовому регістрі B. На наступному етапі воно подається на порт запису даних (WD) пам'яті. Новий керуючий сигнал MemWrite показує, коли саме дані повинні бути записані в пам'ять.

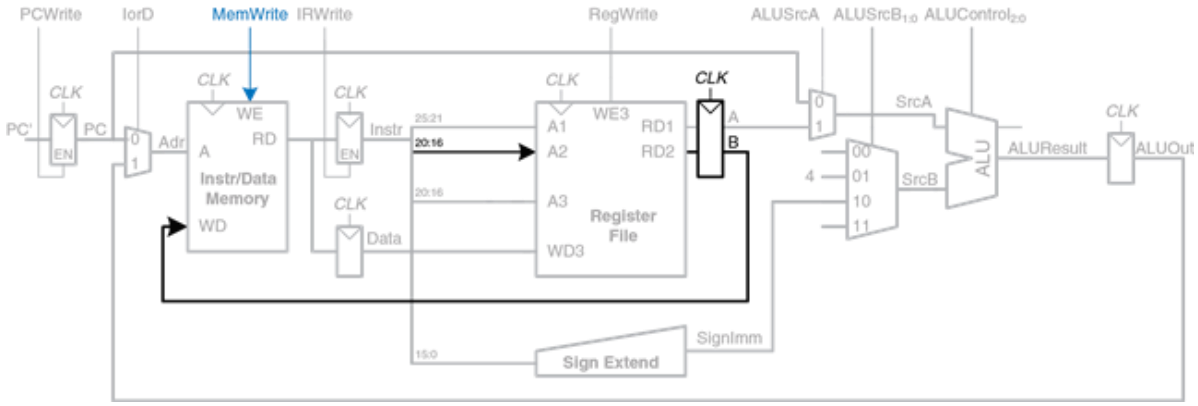

Рис. 20

У випадку команд типу R відразу після вибірки команди з пам'яті з регістрового файлу читаються два операнда. Сигнал ALUSrcB1: 0, керуючий мультиплексором SrcB, дозволяє вибрати регістр B в якості другого операнда АЛП, як показано на Рис. 21. АЛП виконує необхідну операцію і зберігає результат в ALUOut. На наступному етапі ALUOut записується назад в регістр,

номер якого вказаний у полі rd (Instr15: 11). Для цього нам будуть потрібні два додаткових мультиплексори. Мультиплексор MemtoReg подає на WD3 або ALUOut (для команд типу R), або Data (для lw). Мультиплексор RegDst дозволяє вибрати, в якому полі команди знаходиться номер регістра, куди буде записаний результат - rt або rd.

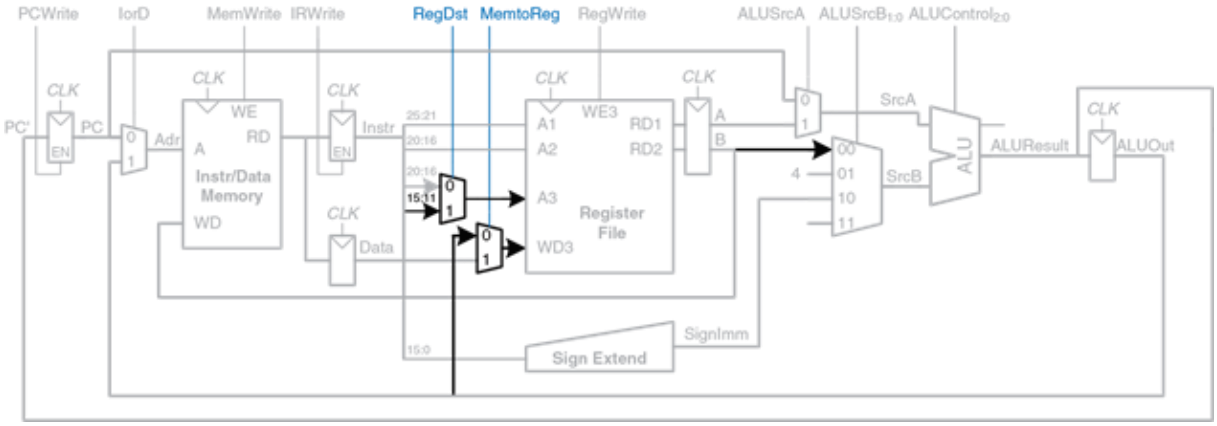

Рис. 21

У випадку команди beq відразу після вибірки команди з пам'яті з регістрового файлу читаються два операнди. Для того щоб визначити, рівні вони чи ні, АЛП віднімає один від іншого і в тому випадку, якщо результат дорівнює нулю, встановлює прапор нуля (Zero flag). Одночасно з цим тракт даних повинен обчислити таке значення лічильника команд на той випадок, якщо умовний перехід буде все-таки виконано: PC '= PC + 4 + SignImm × 4. У однотактному процесорі для цього використовувався ще один суматор. Тепер же ми можемо використовувати АЛП в третій раз. Таким чином, спочатку АЛП вирахує значення PC + 4 і запише його в лічильник команд, також як і для всіх інших команд. Потім АЛП використовує оновлене значення PC для обчислення PC + SignImm × 4. Для множення SignImm на чотири як і раніше, просто виконаємо зсув вліво на два розряди, як показано на Рис. 22. Мультиплексор SrcB вибирає зсунуте значення, після чого воно підсумовується з PC, в результаті чого виходить адреса переходу, яка і зберігається в ALUOut. Ще один новий мультиплексор, керований сигналом PCSrc, вибирає один із сигналів і подає його на PC '. Лічильник команд оновлюється, якщо встановлений сигнал PCWrite, або якщо виконаний умовний перехід. Керуючий сигнал Branch показує, чи є команда, що виконується, командою умовного переходу beq. Умовний перехід буде виконано, якщо встановлений прапор нуля. Таким чином, сигнал дозволу запису в лічильник команд PCEn дорівнює одиниці або тоді, коли встановлений сигнал PCWrite, або коли сигнал Branch встановлений одночасно з прапором нуля.

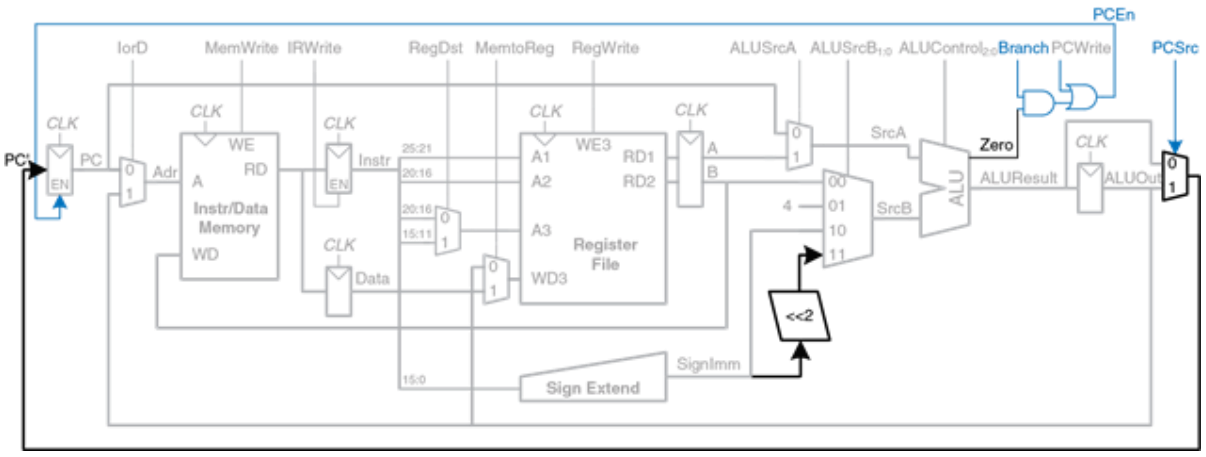

Рис. 22

2. Багатотактний пристрій управління

Пристрій управління формує керуючі сигнали в залежності від полів opcode і funct команди (Instr31: 26 і Instr5: 0 відповідно).

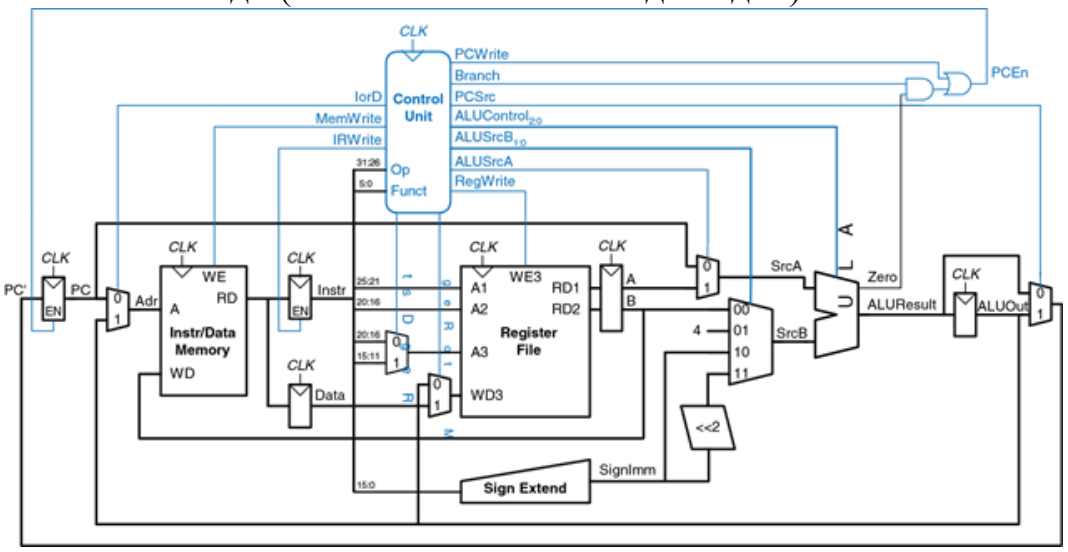

Рис. 23

Як і в однотактному процесорі, пристрій управління поділено на дві частини, як показано на Рис. 24. Дешифратор АЛП залишається точно таким же, як і раніше.

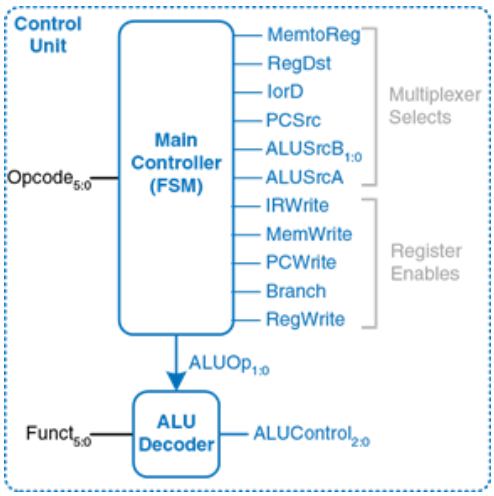

Рис.24

А ось замість основного дешифратора нам знадобиться кінцевий автомат, так як керуючі сигнали тепер будуть залежати не тільки від того, яка команда виконується в даний момент, але і від того, на якому етапі виконання вона знаходиться. Будемо називати цей автомат керуючим автоматом.

Керуючий автомат формує сигнали управління мультиплексорами і сигнали дозволу запису в регістри тракту даних. Сигнали для мультиплексорів називаються MemtoReg, RegDst, IorD, PCSrc, ALUSrcB і ALUSrcA. Сигнали дозволу запису називаються IRWrite, MemWrite, PCWrite, Branch і RegWrite.