# Архитектура вычислительных систем

Лекция 11.

Яревский Е.А.

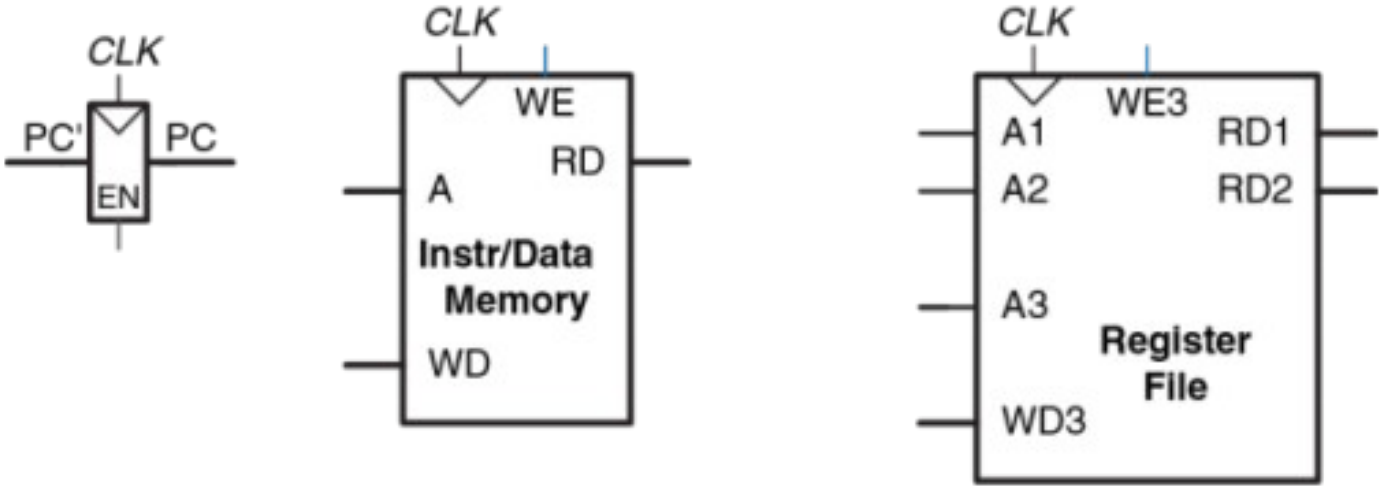
Кафедра вычислительной физики

# МНОГОТАКТНЫЙ ПРОЦЕССОР

Выполнение каждой команды разбивается на несколько этапов.
На каждом этапе процессор может читать или писать данные в память или регистровый файл, или выполнять какую-нибудь операцию в АЛУ.
У разных команд – разное количество этапов, так что простые команды выполняются быстрее, чем сложные.
Один сумматор, на разных этапах он используется для разных целей.
Общая памятью для команд и данных: команды выбираются на первом этапе, а чтение/ запись данных происходит на одном из последующих этапов.

Сконструируем тракт данных, соединяя при помощи комбинационной логики блоки памяти и блоки, хранящие архитектурное состояние процессора.
Необходимо добавить **дополнительные блоки** для хранения информации о **промежуточном (неархитектурном) состоянии** между этапами.

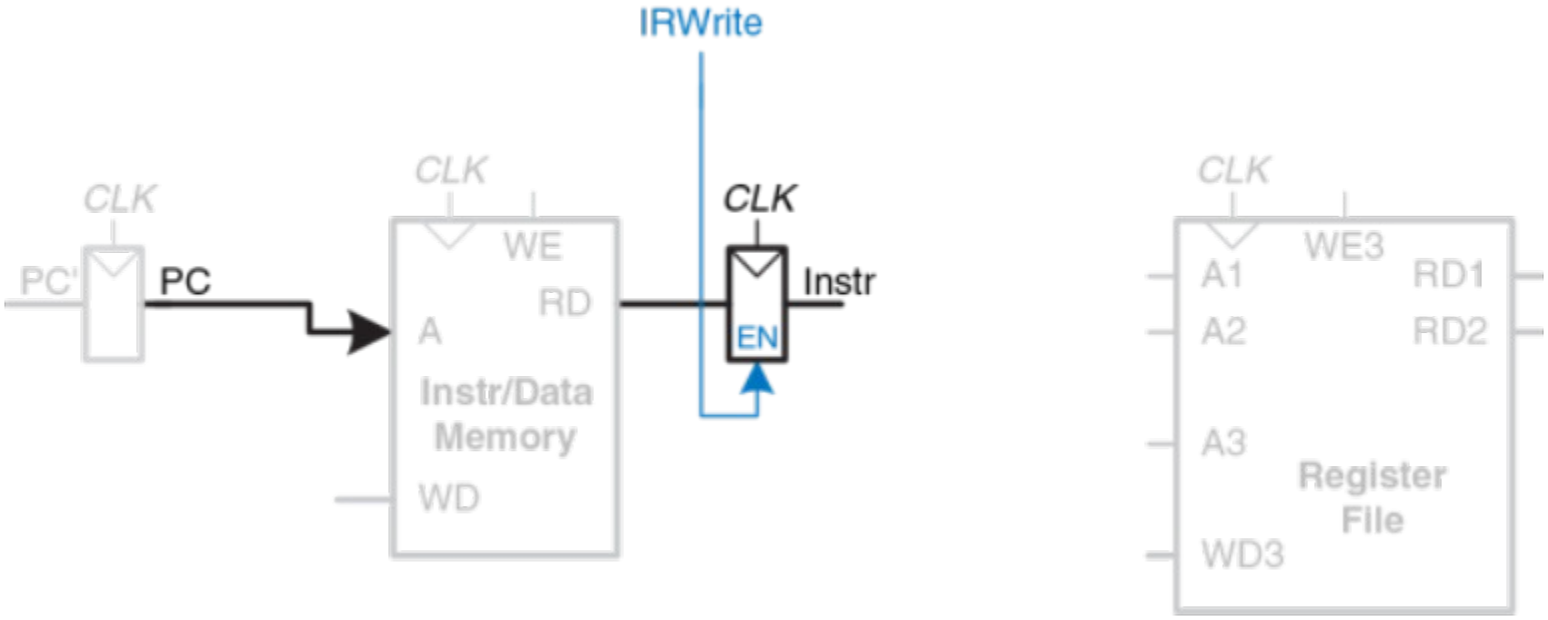
Устройство управления будет должно формировать разные управляющие сигналы в зависимости от текущего этапа выполнения команды, так что вместо комбинационных схем понадобится конечный автомат.




Общая память команд/данных и элементы, хранящие состояние процессора.

Будем использовать общую память, хранящую и команды, и данные.
Это возможно благодаря тому, что теперь можно выбирать команду на одном такте, а обращаться к памяти данных на другом.
Счетчик команд и регистровый файл не меняются.

**Первый шаг**: чтение команды из памяти (выборка).




Выборка команды из памяти.

Прочитанная из памяти команда сохраняется во временный (неархитектурный) регистр команд (Instruction Register), и может использоваться в следующих тактах. Сигнал разрешения записи в регистр команд – IRWrite – будет использоваться, когда потребуется обновить находящуюся в регистре команду.

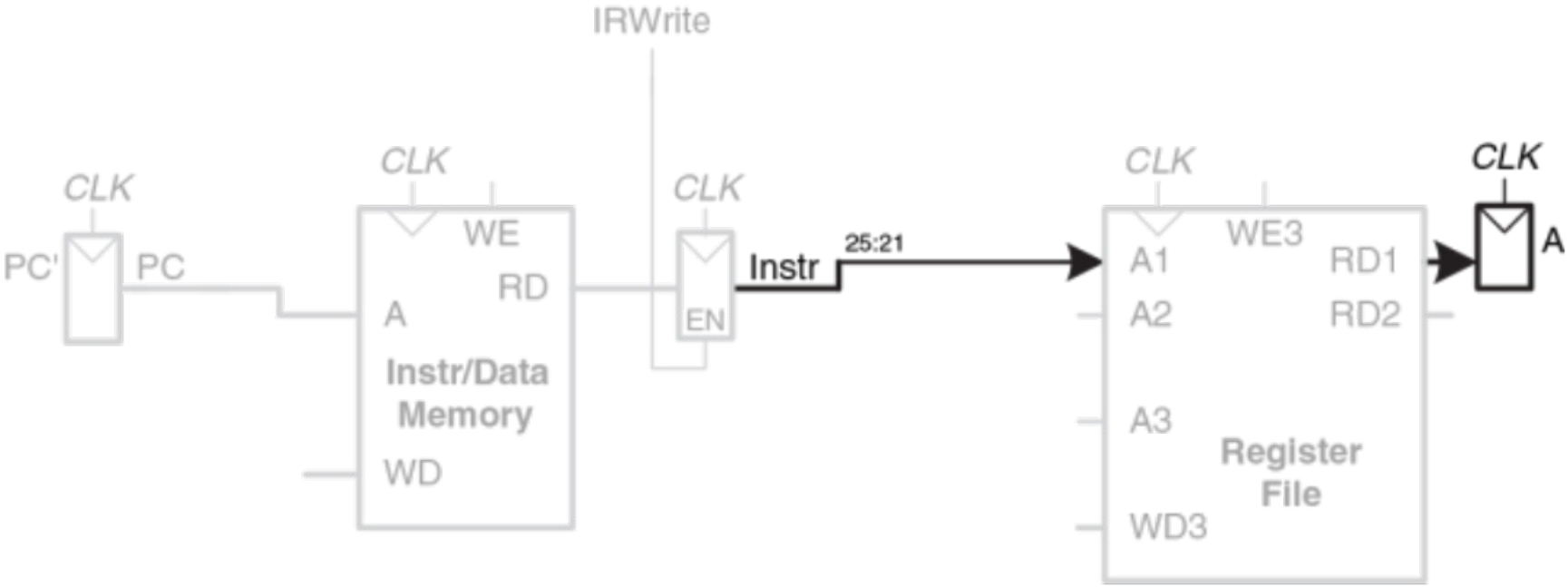
Начнем с команды lw.
Для команды lw следующим шагом будет чтение регистра, содержащего базовый адрес.

# Многотактный тракт данных

Начнем с команды lw.

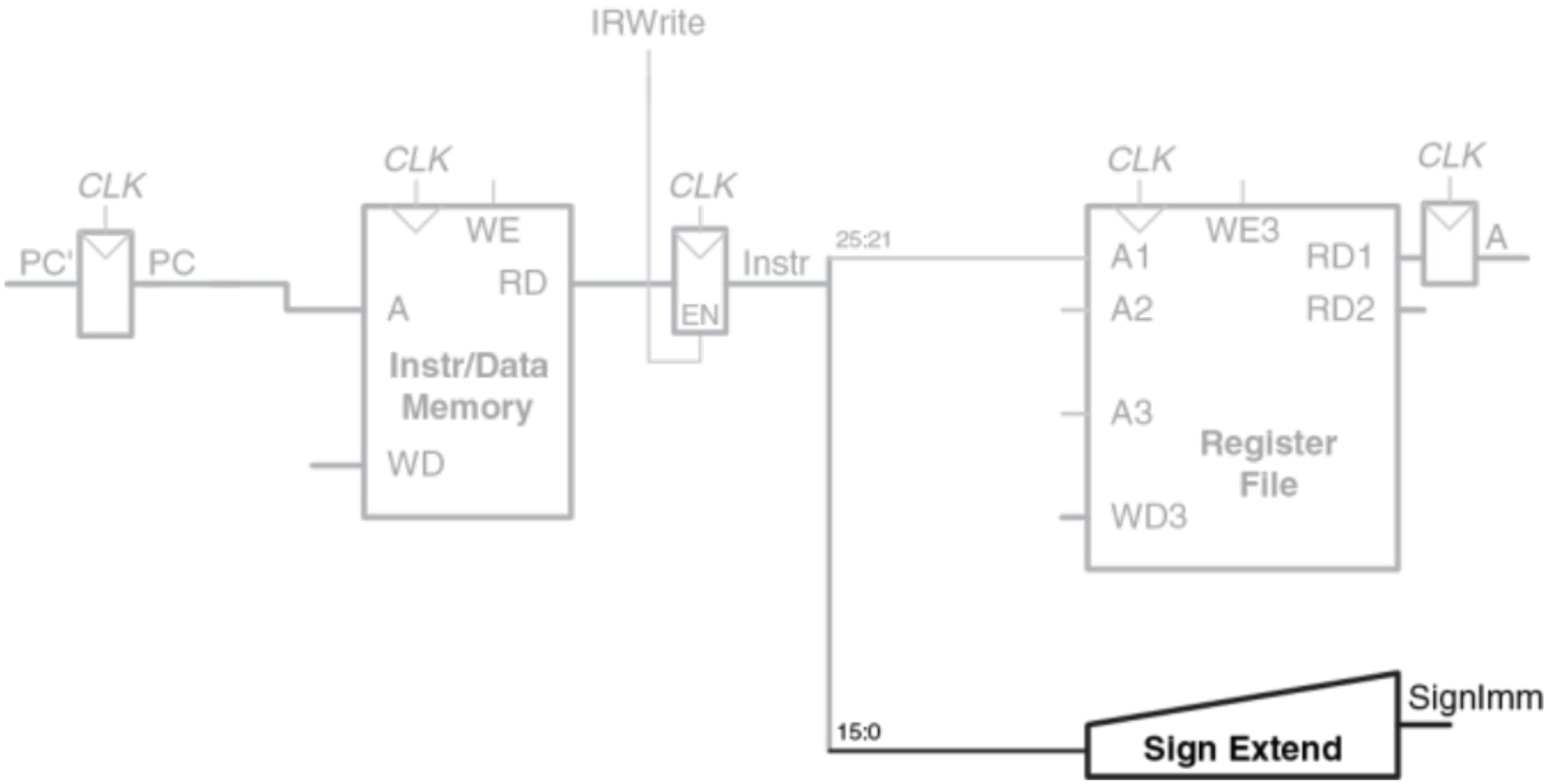
Для команды lw следующим шагом будет чтение регистра, содержащего базовый адрес.

Номер регистра указывается в поле rs (Instr[25:21]) и подается на адресный вход первого порта (A1) регистрового файла. Значение, прочитанное из регистрового файла, появляется на его выходе RD1, и сохраняется во временный регистр A.



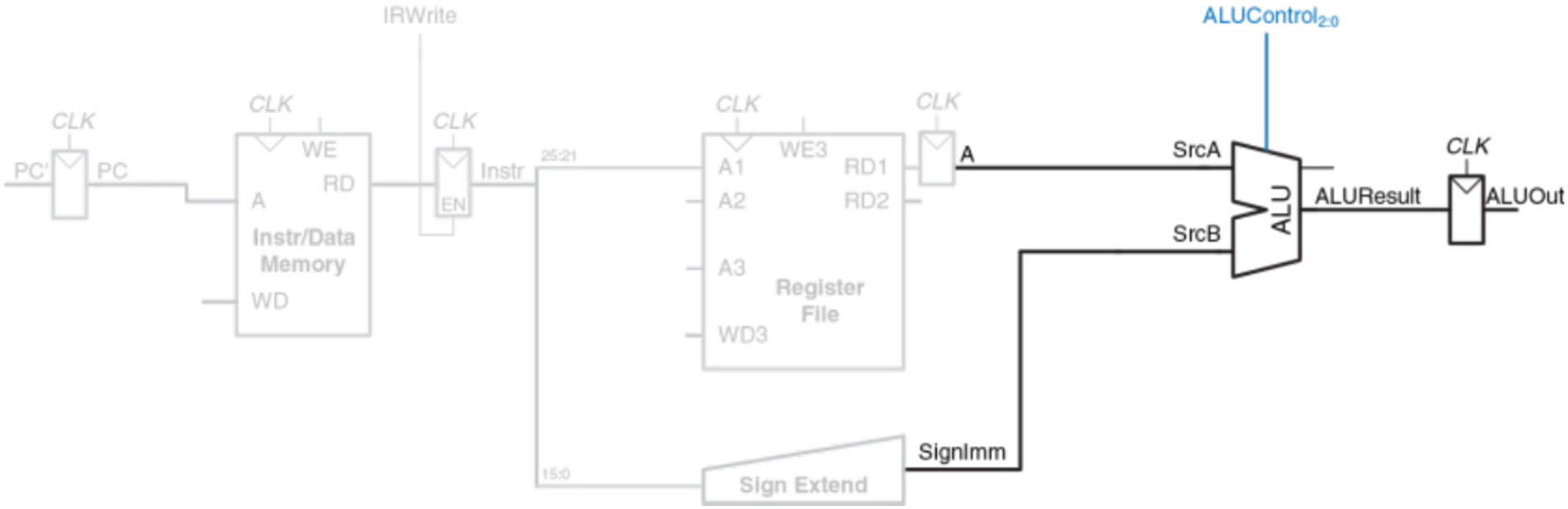
# Многотактный тракт данных

Команде lw также требуется смещение, которое будет прибавлено к базовому адресу и передается как непосредственный операнд в поле Instr[15:0].
Над ним должна быть выполнена операция знакового расширения до 32 бит.
SignImm[15:0] = Instr[15:0], SignImm[31:16] = Instr[15].
Добавлять еще один временный регистр для хранения SignImm смысла нет – это выход комбинационной схемы, вход которой зависит исключительно от Instr.



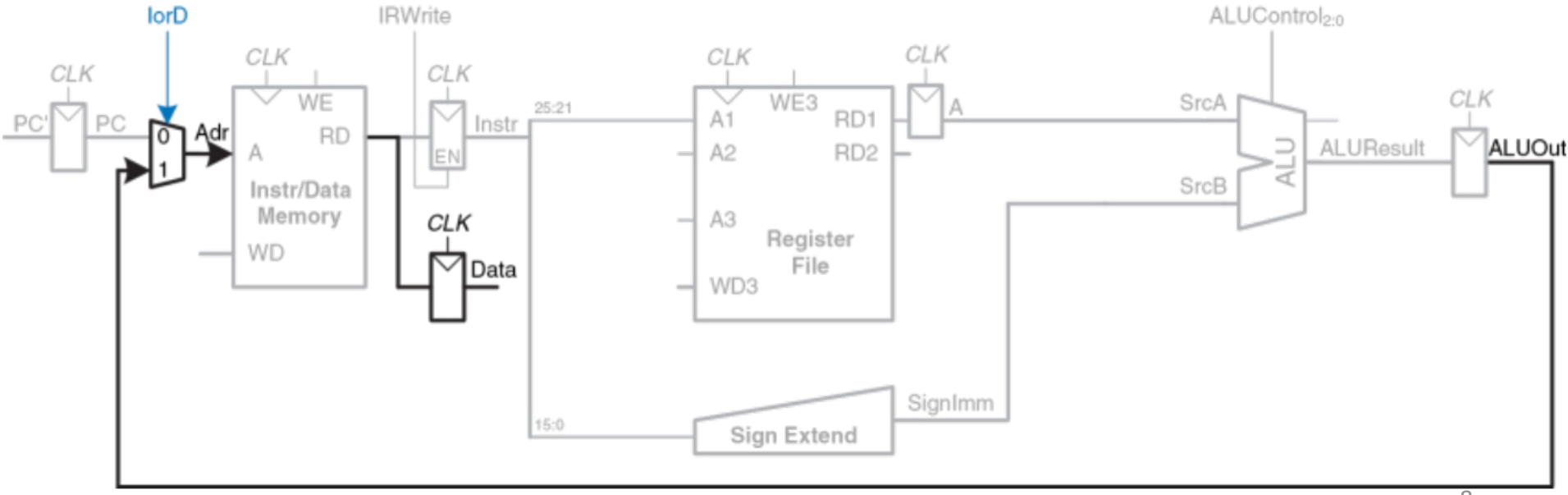
# Многотактный тракт данных

Адрес чтения из памяти получается путем сложения базового адреса и смещения.
Для сложения используется АЛУ.
Чтобы АЛУ выполнило сложение, управляющий сигнал ALUControl должен быть равен 010.
ALUResult сохраняется во временном регистре ALUOut.



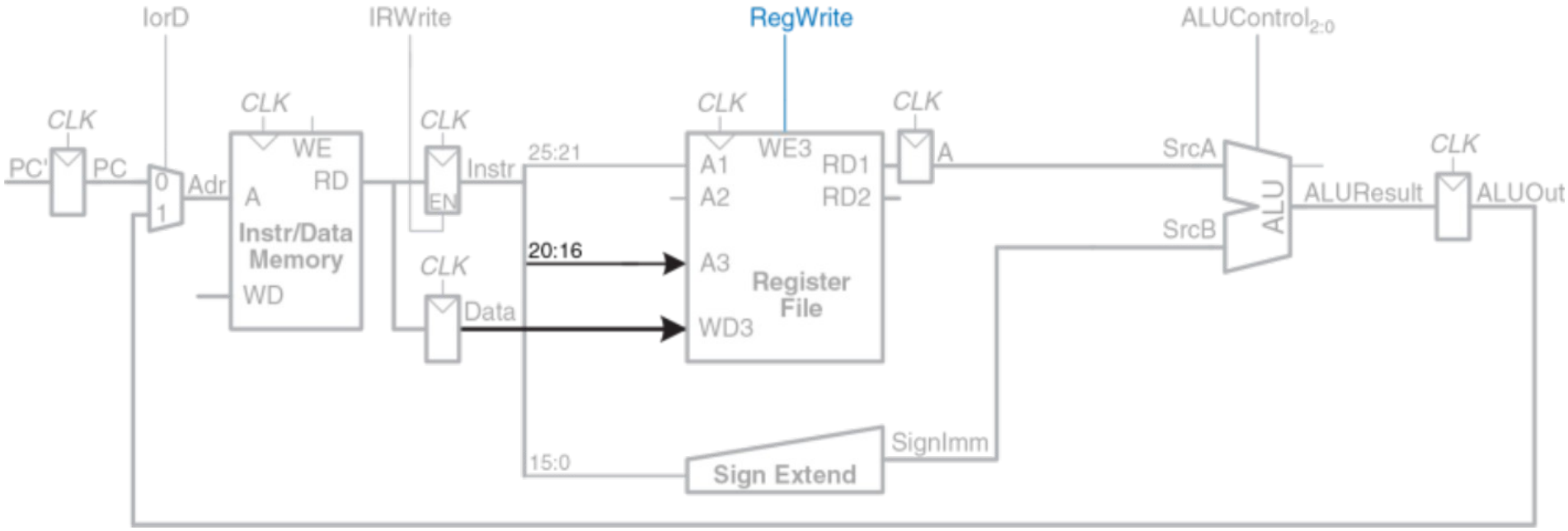
# Многотактный тракт данных

**Следующий шаг**: чтение данных из памяти по вычисленному адресу.
Для этого перед адресным входом памяти необходимо добавить мультиплексор, чтобы в качестве адреса Adr можно было использовать либо PC, либо ALUOut
Этот мультиплексор управляется сигналом IorD – выбор адреса команд/данных.
Прочитанные из памяти данные сохраняются во временном регистре Data.
Мультиплексор адреса позволяет повторно использовать память во время выполнения команды lw.
Сначала в качестве адреса используется PC: выборка команды.
Затем в качестве адреса используется ALUOut: читаются данные.
Управляющий сигнал IorD должен принимать разные значения на разных этапах выполнения команды.



# Многотактный тракт данных

Данные должны быть записаны в регистровый файл.
Номер регистра результата определяется полем rt (Instr[20:16]).



# Многотактный тракт данных

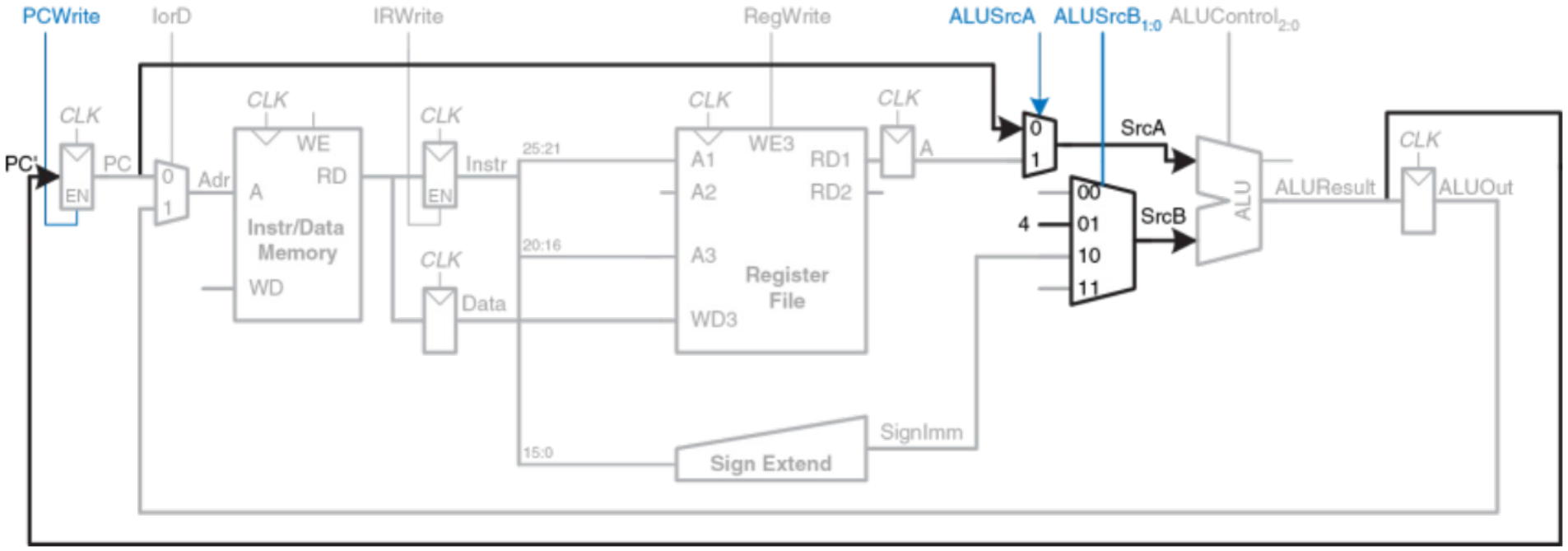
Пока выполняются все перечисленные операции, процессор должен увеличить счетчик команд на четыре.

В МП можно использовать уже имеющееся АЛУ на одном из первых этапов, пока оно еще не используется для других действий.

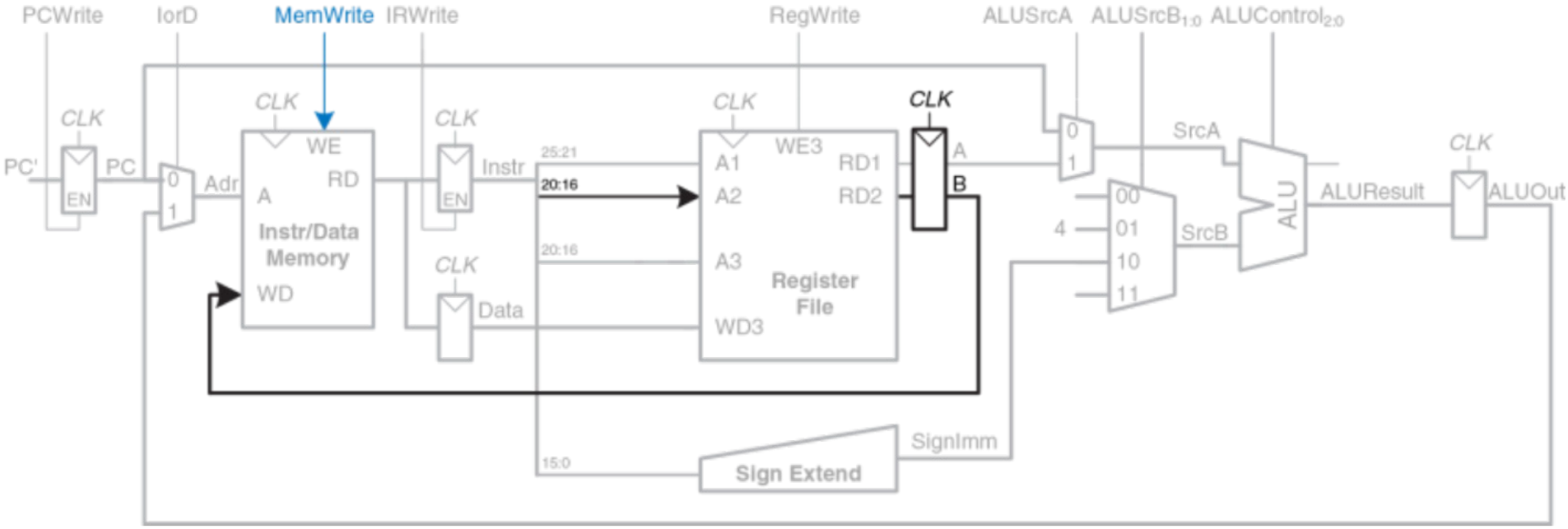
Двухвходовой мультиплексор (2:1), управляемый сигналом ALUSrcA, подает на SrcA либо PC, либо регистр A.

Четырехвходовой мультиплексор (4:1), управляемый сигналом ALUSrcB, подает на SrcB либо константу 4, либо SignImm.

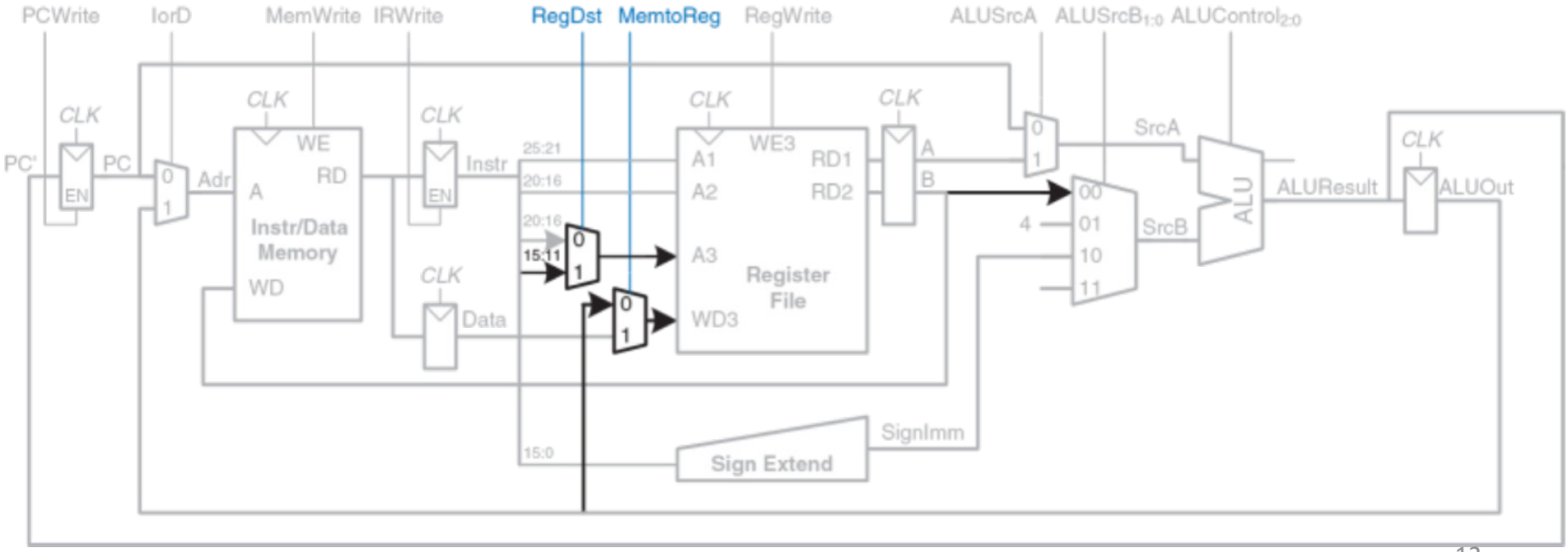
Для обновления счетчика команд, АЛУ складывает SrcA (PC) и SrcB (4) и записывает результат в PC. Управляющий сигнал PCWrite разрешает запись в PC только на тех тактах, где это необходимо.



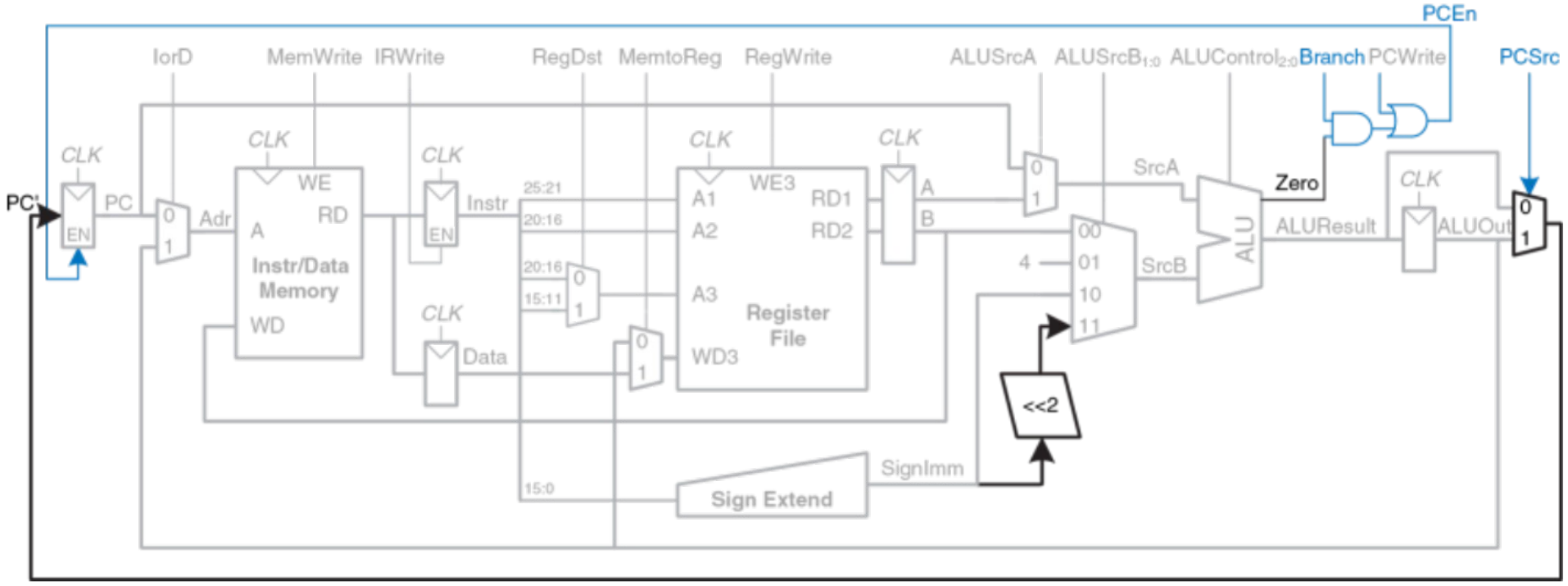
Добавим поддержку **команды sw.**
Как lw, sw читает базовый адрес из первого порта РФ, выполняет знаковое расширение, затем АЛУ складывает их, получая адрес для записи в память.
Отличие sw: нужно прочитать еще один регистр из РФ и записать его содержимое в память. Номер регистра указан в поле rt (Instr[20:16]), которое подключено ко второму порту РФ.
Прочитанное значение сохраняется во временном регистре B.
На следующем шаге оно подается на порт записи данных (WD) памяти.
Новый управляющий сигнал MemWrite показывает, когда именно данные должны быть записаны в память.



**Поддержка команд типа R**.
После выборки команды из памяти, из РФ читаются два операнда.
Сигнал ALUSrcB1:0, управляющий мультиплексором SrcB, позволяет выбрать регистр B в качестве второго операнда АЛУ.
АЛУ выполняет требуемую операцию и сохраняет результат в ALUOut.
На следующем этапе ALUOut записывается обратно в регистр, номер которого указан в поле rd (Instr[15:11]).
Мультиплексор MemtoReg подает на WD3 или ALUOut (команды R), или Data (lw).
Мультиплексор RegDst выбирает, в каком поле команды находится номер регистра для записи результата – rt или rd.



**Поддержка команды beq**.
После выборки команды из регистрового файла читаются два операнда.
АЛУ вычитает их и устанавливает флаг нуля (Zero flag) при равенстве.
Одновременно с этим тракт данных должен вычислить следующее значение PC на случай, если условный переход нужно выполнять PC′ = PC + 4 + SignImm × 4.
Можем использовать АЛУ в третий раз.



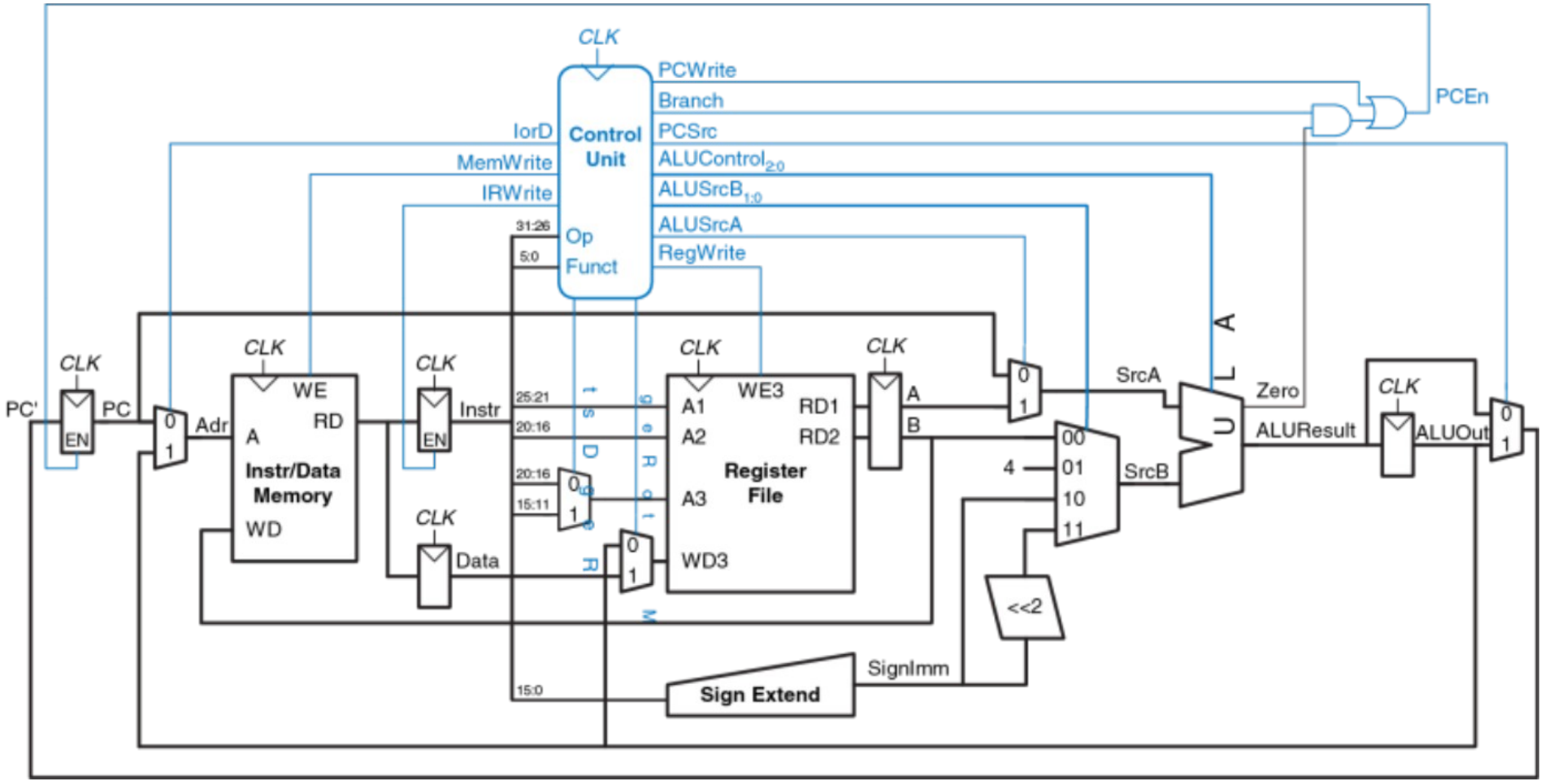
# Многотактный тракт данных

Сначала АЛУ вычислит значение PC + 4 и запишет его в счетчик команд, также как для всех других команд.
Затем АЛУ использует обновленное значение PC для вычисления PC+SignImm×4.
Для умножения SignImm на четыре, оно сдвигается влево на два разряда.
Мультиплексор SrcB выбирает сдвинутое значение, после чего оно суммируется с PC, в результате чего получается адрес перехода, который сохраняется в ALUOut.
Мультиплексор, управляемый сигналом PCSrc, выбирает один из сигналов и подает его на PC′. Счетчик команд обновляется, если установлен сигнал PCWrite, или если выполнен условный переход. Управляющий сигнал Branch показывает, является ли выполняющаяся команда командой условного перехода beq.
Условный переход будет выполнен, если установлен флаг нуля.
Т.о., сигнал разрешения записи в счетчик команд PCEn равен единице
или тогда, когда установлен сигнал PCWrite,
или когда сигнал Branch установлен одновременно со флагом нуля.

Многотактный тракт данных описан.
Потребовались невидимые программисту временные (неархитектурные) регистры, чтобы сохранять результаты каждого из этих этапов.
Смогли повторно использовать одно и то же АЛУ (избавились от нескольких сумматоров).
Смогли поместить команды и данные в общую память.

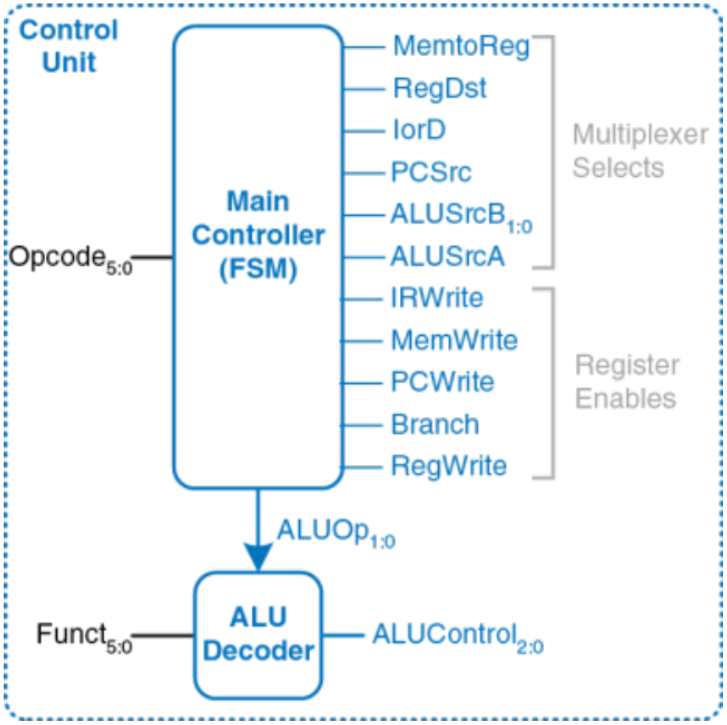
# Многотактное устройство управления

Устройство управления формирует управляющие сигналы в зависимости от полей opcode и funct команды (Instr[31:26] и Instr[5:0], соответственно).



# Многотактное устройство управления

УУ поделено на две части, как и раньше.
Дешифратор АЛУ остается точно таким же, как и в однотактном случае.




Вместо основного дешифратора понадобится конечный автомат, так как управляющие сигналы теперь зависят не только от выполняемой команды, но и от того, на каком этапе выполнения она находится.
Этот автомат – **управляющий автомат**.
Нужно разработать его диаграмму состояний.

УА формирует сигналы управления мультиплексорами и сигналы разрешения записи в регистры тракта данных.

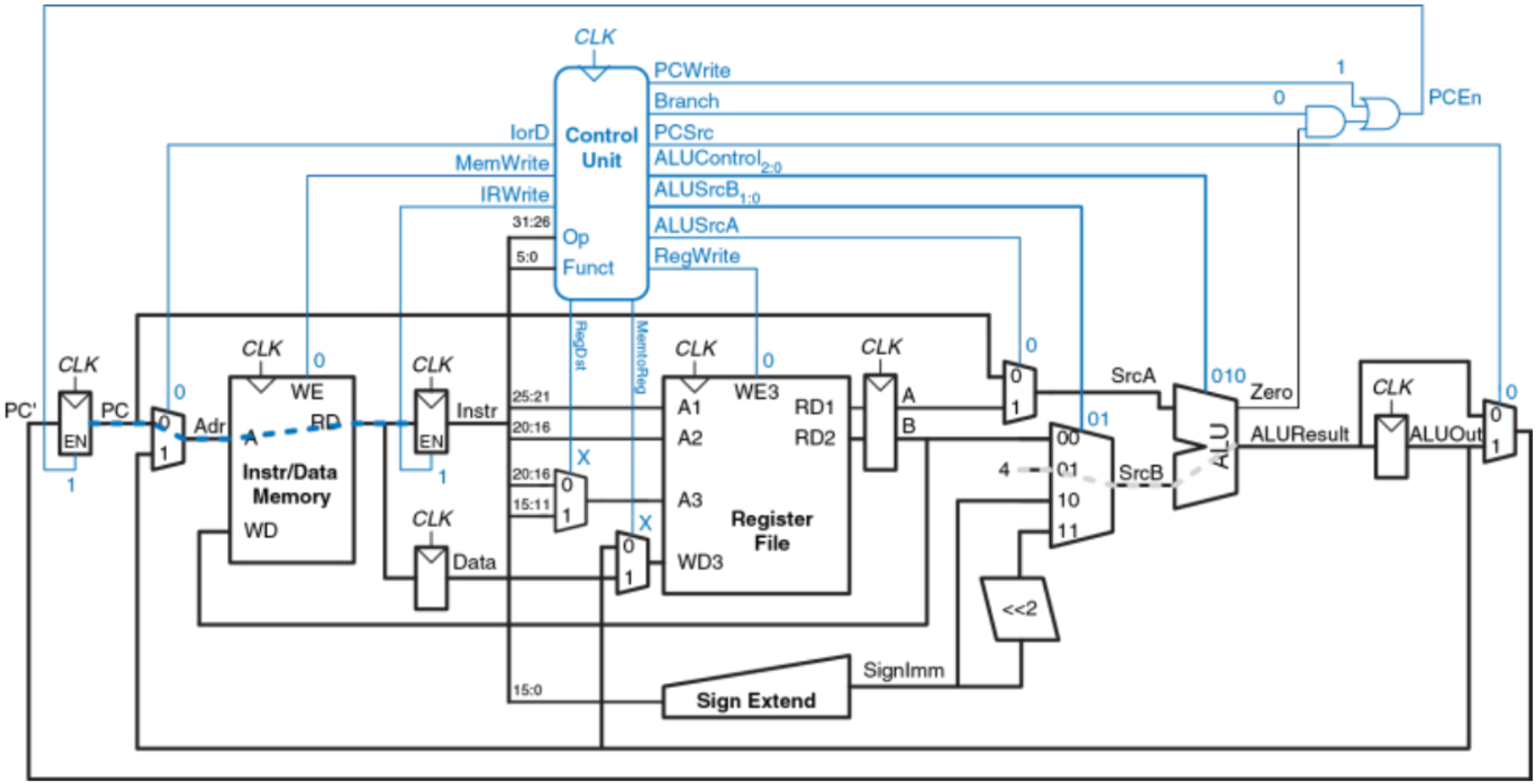
Указываются только те управляющие сигналы, которые имеют смысл на конкретном этапе выполнения команды.
Сигналы управления мультиплексорами указываются тогда, когда они действительно используются.

# Многотактное устройство управления

Первый этап команды: чтение из памяти по адресу, находящемуся в PC (выборка команды из памяти).
В это состояние управляющий автомат переходит по сигналу сброса (reset).
Чтобы адрес для чтения был взят из счетчика команд, IorD == 0.
Чтобы прочитанное значение попало в регистр команд (IR), IRWrite == 1.
Одновременно с этим PC должен быть увеличен на четыре – после этого он будет указывать на следующую команду.
АЛУ в этот момент свободно, и процессор может использовать его для вычисления PC + 4 одновременно с выборкой команды из памяти.
ALUSrcA = 0, и на первый вход АЛУ (SrcA) подается PC.
ALUSrcB = 01, и на второй вход АЛУ (SrcB) подается константа 4.
ALUOp = 00, и дешифратор АЛУ устанавливает ALUControl == 010, чтобы АЛУ выполнило сложение.


Чтобы содержимое счетчика команд обновилось, нужно, чтобы PCSrc == 0, а PCWrite == 1.

Пути движения данных на этапе выборки команды.



# Многотактное устройство управления

Полная диаграмма состояний управляющего автомата.

# Многотактное устройство управления

S1) На этом этапе происходит чтение из регистрового файла и дешифрация команды. Из регистрового файла всегда читаются два регистра, номера которых указаны в полях rs и rt команды Instr. Одновременно с этим происходит знаковое расширение непосредственного операнда. Дешифрация команды заключается в том, что процессор смотрит в поле opcode, чтобы понять, что от него требуется.

S2) Далее, в зависимости от поля opcode, управляющий автомат должен перейти в одно из нескольких состояний. Если выполняется команда загрузки или сохранения данных (lw или sw), процессор вычисляет адрес в памяти путем сложения базового адреса и непосредственного операнда с расширенным знаком. ALUSrcA = 1, чтобы в качестве первого операнда был выбран регистр A. ALUSrcB = 10, чтобы в качестве второго операнда был выбран сигнал SignImm. ALUOp = 00, поэтому АЛУ выполнит сложение. Полученный адрес сохраняется в регистр ALUOut для использования на следующем этапе.

S3,S4) Далее, в случае команды lw многотактный процессор должен прочитать данные из памяти и сохранить их в регистровый файл. Для чтения данных из памяти сигнал IorD==1, в этом случае адрес будет взят из регистра ALUOut. Прочитанное значение сохраняется в регистр Data (S3). На этапе (S4), содержимое Data записывается в регистровый файл, для чего сигнал MemtoReg==1, а RegDst==0. Для завершения записи управляющий автомат устанавливает сигнал RegWrite. Наконец, автомат возвращается в первоначальное состояние S0, чтобы выбрать из памяти следующую команду.

# Особенности многотактного процессора

Многотактному процессору для выполнения разных команд требуется разное число тактов. Поскольку он выполняет меньшее количество действий за такт, то его такты гораздо короче, чем у однотактного.

На каждом такте процессор выполняет либо арифметическую операцию в АЛУ, либо операцию доступа к памяти, либо операцию доступа к регистровому файлу (самые медленные операции).

Для команд beq и j многотактному процессору нужно три такта, для команд sw, addi и команд типа R – четыре, а для команды lw – пять.
Суммарное число тактов на команду (CPI) будет зависеть от частоты использования каждой из команд.

Например, на тестовом наборе SPECINT2000 CPI = 4.12.
(если считать по самой длинной команде: CPI = 5).
Длительность такта на 65-нм КМОП техпроцессе: **325 пс**.
Однотактный: **925 пс**.

# Особенности многотактного процессора

Многотактный процессор оказался **медленнее**, чем однотактный!

Фундаментальная проблема оказалась в том, что, несмотря на разбиение самой медленной команды (lw) на пять этапов, длительность такта в многотактном процессоре уменьшилась вовсе не в пять раз.
Одной из причин явилось то, что не все этапы стали одинаковой длины.

Другой причиной стали накладные расходы, связанные со временными регистрами – задержка в 50 пс, равная сумме времени предустановки и времени срабатывания регистра, теперь добавляется не к общему времени выполнения команды, а к каждому этапу по отдельности.

В сравнении с однотактным процессором многотактный процессор, скорее всего, окажется меньшего размера, так как нет необходимости в двух дополнительных сумматорах, а память команд и данных объединена в один блок. Взамен ему требуется пять неархитектурных (временных) регистров и несколько дополнительных мультиплексоров.