

Основы компьютерной техники (Computer Organization. Basis)

БГТУ

кафедра ПИ

доцент Самаль Дмитрий Иванович т. 293-23-79, [dmitry\_samal@mail.ru,](mailto:dmitry_samal@mail.ru) a.510б-5

Лекция 8

«УУ однотактного процессора» 2020



### План лекции

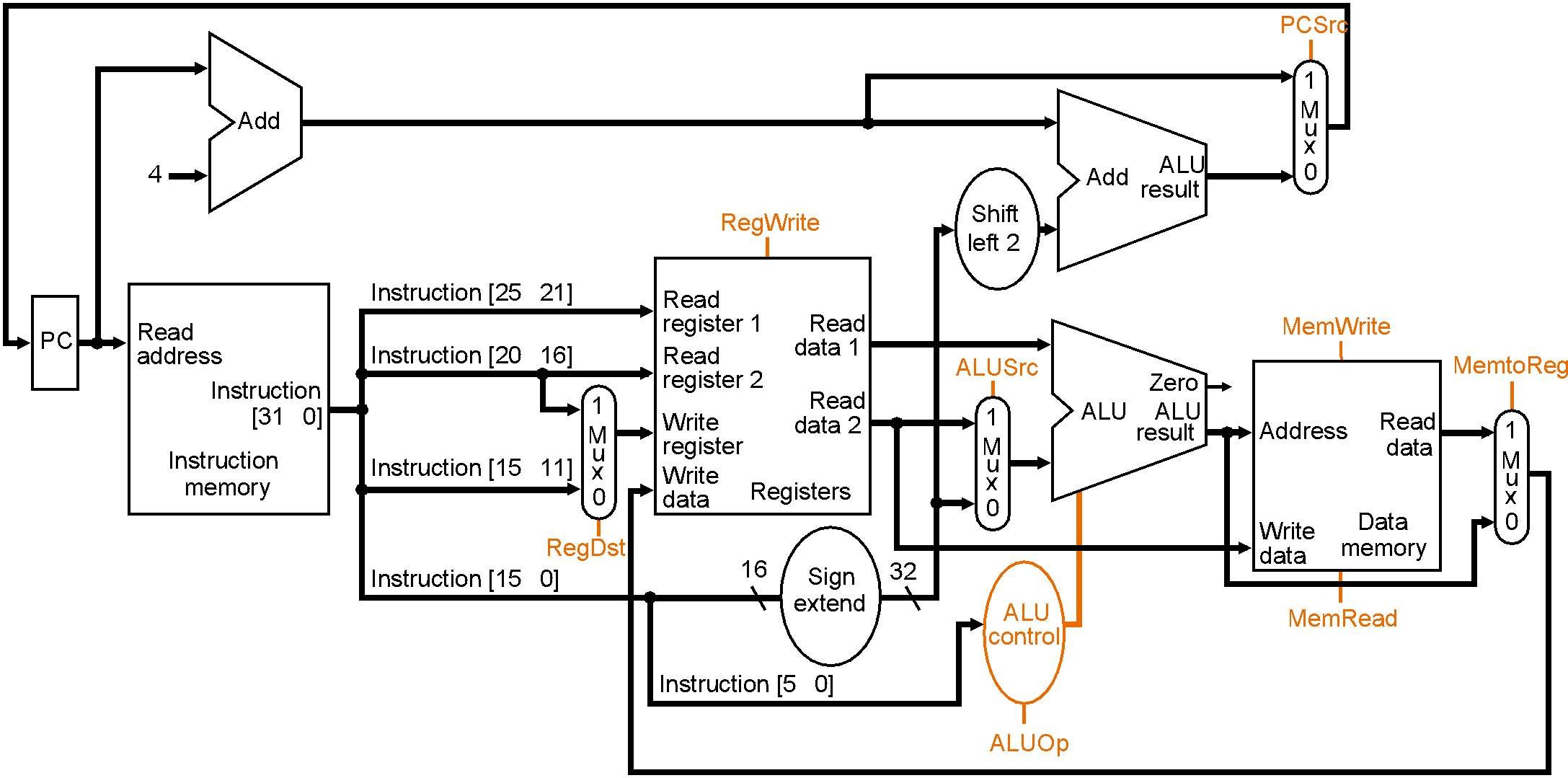
1. Устройство управления однотактного процессора

***Слайд*** 2



# Собирая всё в одну схему

* Осталось только сгенерировать управляющие сигналы



***Слайд*** 3

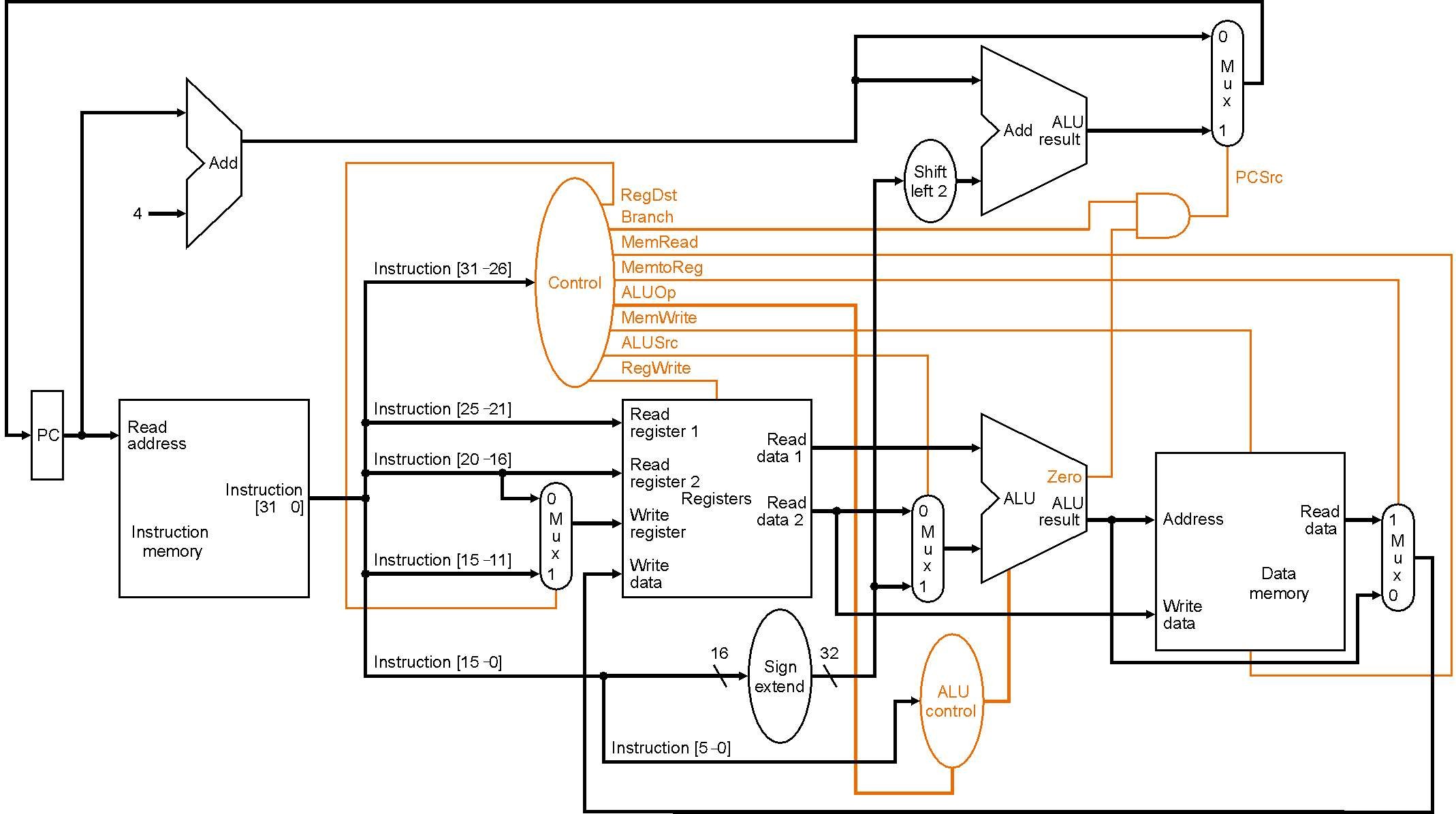


# Устройство управления однотактного процессора

## (Control Logic for the Single-Cycle CPU)



Добавление управляющих сигналов



***Слайд*** 5



# Управляющие биты АЛУ

* + 5 функций АЛУ:

|  |  |  |
| --- | --- | --- |
| **ALU control input** | **Function** | **Operations** |
| 000 | And | and |
| 001 | Or | or |
| 010 | Add | add, lw, sw |
| 110 | Subtract | sub, beq |
| 111 | Slt | slt |

***Слайд*** 6

# Управляющие биты АЛУ

* + 5 функций АЛУ:

|  |  |  |
| --- | --- | --- |
| **ALU control input** | **Function** | **Operations** |
| 000 | And | and |
| 001 | Or | or |
| 010 | Add | add, lw, sw |
| 110 | Subtract | sub, beq |
| 111 | Slt | slt |

* + коды берутся из КОП (биты 31-26), а так же из поля function (биты 5-0)
  + АЛУ нет необходимости сообщать все коды операций – учитываются только имеющие отношение к ALU (2 bits):

00 - lw,sw 01 - beq 10 - R-format

func

**ALU**

**Control**

**Main Control**

op 6



6 ALUop

2

ALUctr

3

***Слайд*** 7



# Generating ALU control

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction**  **opcode** | **ALUOp** | **Instruction**  **operation** | **Function**  **code** | **Desired**  **ALU**  **action** | **ALU**  **control input** |
| lw | 00 | load word | xxxxxx | add | 010 |
| sw | 00 | store word | xxxxxx | add | 010 |
| beq | 01 | branch eq | xxxxxx | subtract | 110 |
| R-type | 10 | add | 100000 | add | 010 |
| R-type | 10 | subtract | 100010 | subtract | 110 |
| R-type | 10 | AND | 100100 | and | 000 |
| R-type | 10 | OR | 100101 | or | 001 |
| R-type | 10 | slt | 101010 | slt | 111 |

ALU

Control Logic

***Слайд*** 8

# Generating individual ALU signals

ALUctr2 = ALUctr1 = ALUctr0 =

|  |  |  |
| --- | --- | --- |
| **ALUop** | **Function** | **ALUCtr signals** |
| 00 | xxxx | 010 |
| 01 | xxxx | 110 |
| 10 | 0000 | 010 |
| 10 | 0010 | 110 |
| 10 | 0100 | 000 |
| 10 | 0101 | 001 |
| 10 | 1010 | 111 |

func

**Main Control**

op 6

6 ALUop

2

ALUctr

3

**ALU**

**Control**

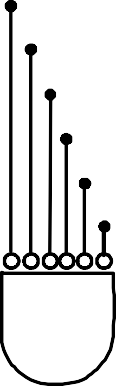
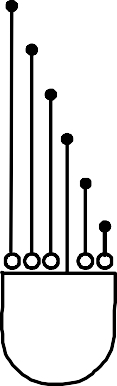


***Слайд*** 9



# Управление

* Простая комбинационная логика (таблица истинности)



Inputs

Op5

Op4 Op3 Op2 Op1

Op0

Outputs

R-format

Iw

sw

beq

RegDst

ALUSrc MemtoReg RegWrite MemRead MemWrite

Branch ALUOp1

ALUOpO

ALUOp

ALU control block

ALUOp0

ALUOp1

F3

Operation2

F2

Operation

F (5– 0)

Operation1

F1

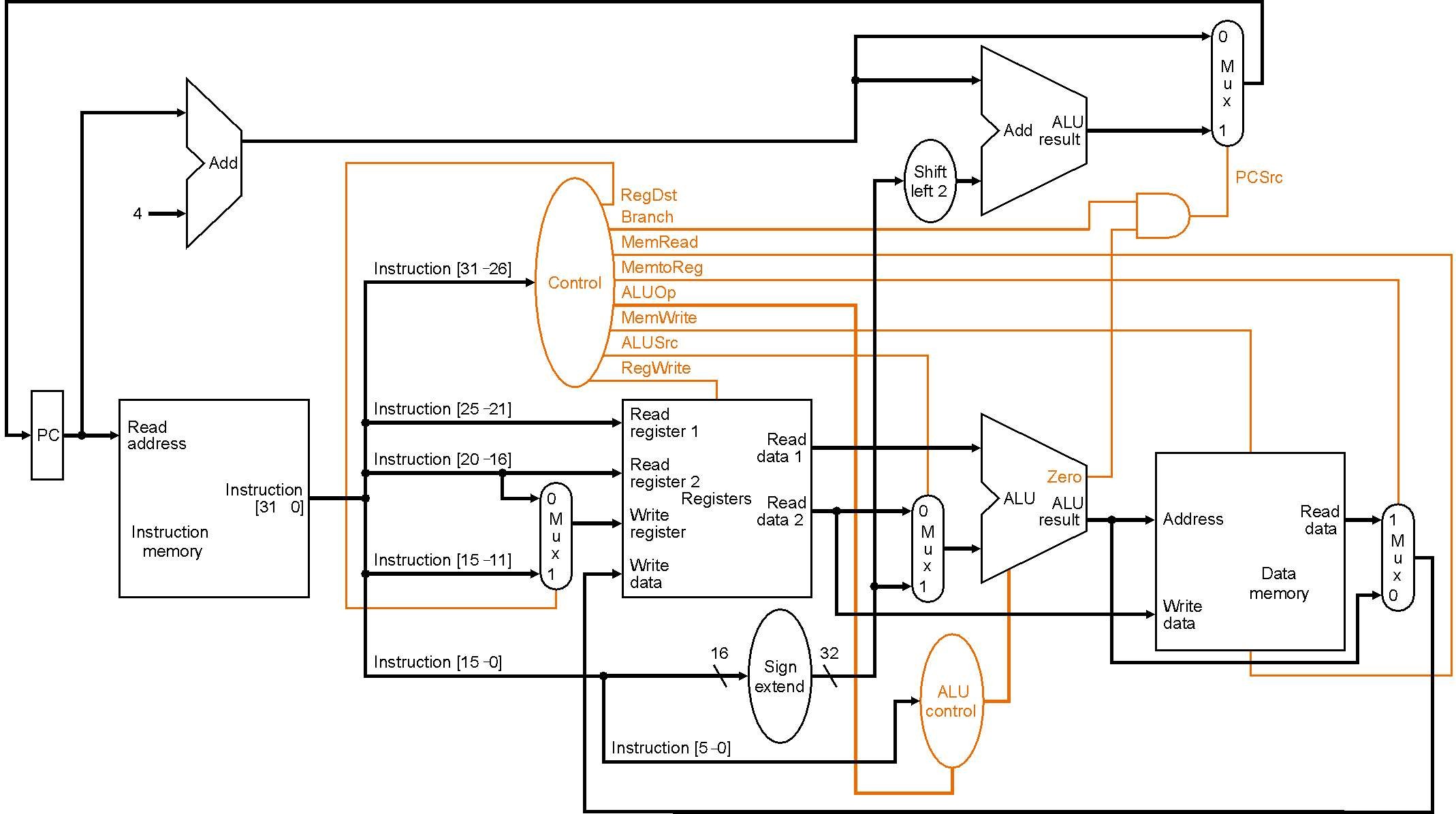
Operation0

F0

***Слайд*** 10



# R-Format Instructions (e.g., Add)

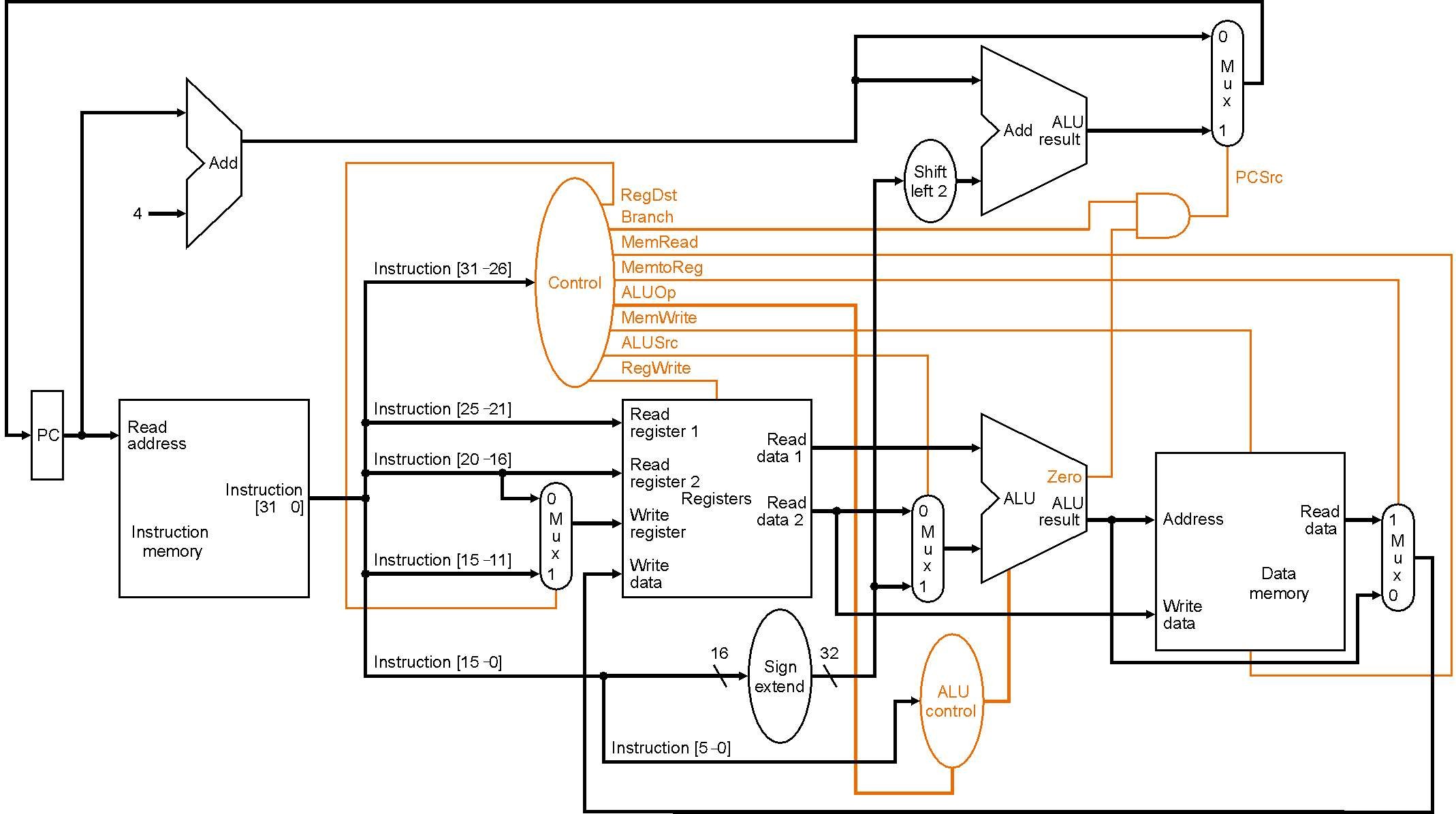


***Слайд*** 11

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **RegDst** | **ALUSrc** | **Memto- Reg** | **Reg Write** | **Mem Read** | **Mem Write** | **Branch** | **ALUOp1** | **ALUp0** |
| R-format |  |  |  |  |  |  |  | 1 | 0 |
| lw |  |  |  |  |  |  |  | 0 | 0 |
| sw |  |  |  |  |  |  |  | 0 | 0 |
| beq |  |  |  |  |  |  |  | 0 | 1 |



# lw управление

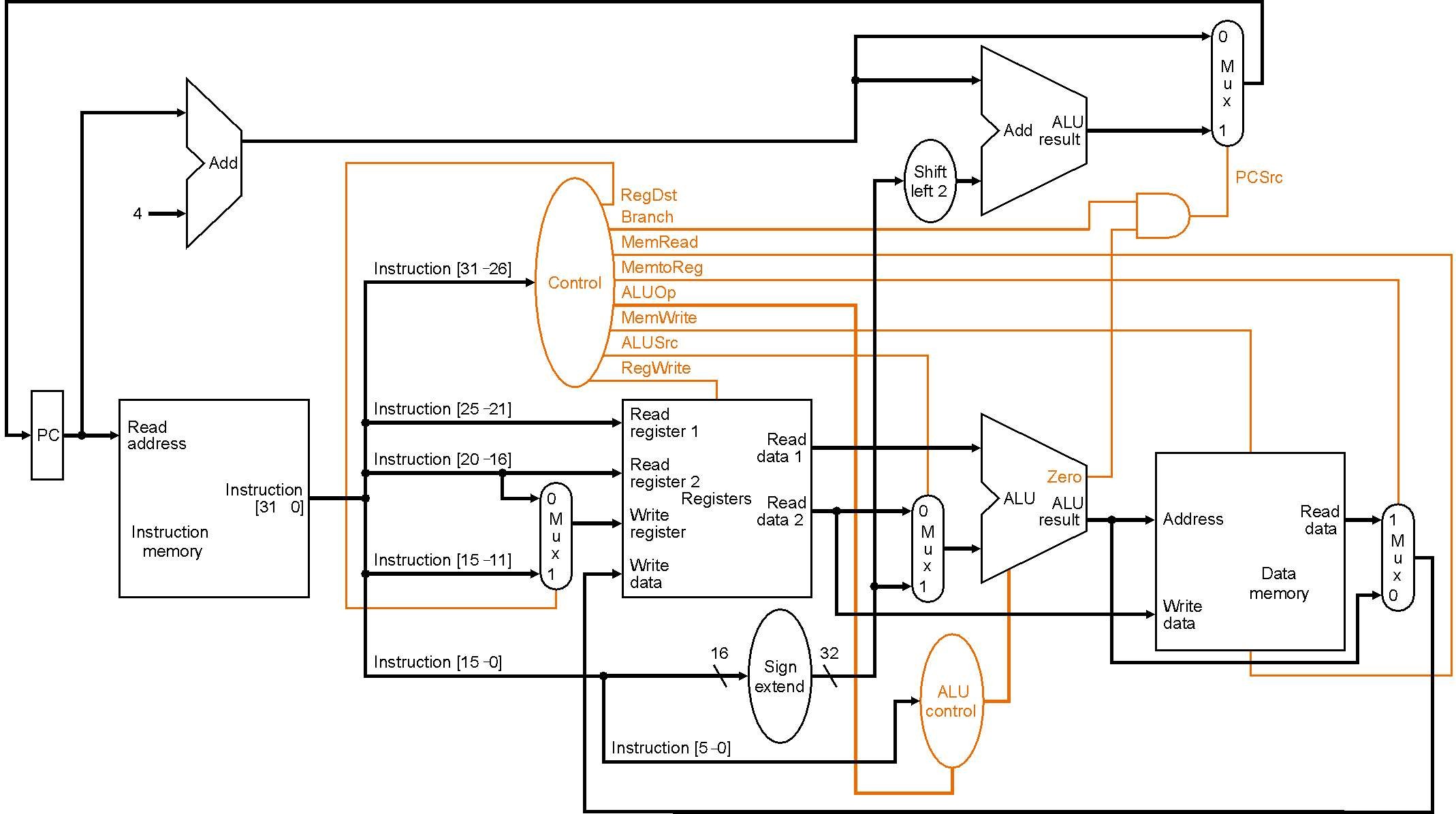


***Слайд*** 12

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **RegDst** | **ALUSrc** | **Memto- Reg** | **Reg Write** | **Mem Read** | **Mem Write** | **Branch** | **ALUOp1** | **ALUp0** |
| R-format | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| lw |  |  |  |  |  |  |  | 0 | 0 |
| sw |  |  |  |  |  |  |  | 0 | 0 |
| beq |  |  |  |  |  |  |  | 0 | 1 |



# sw управление

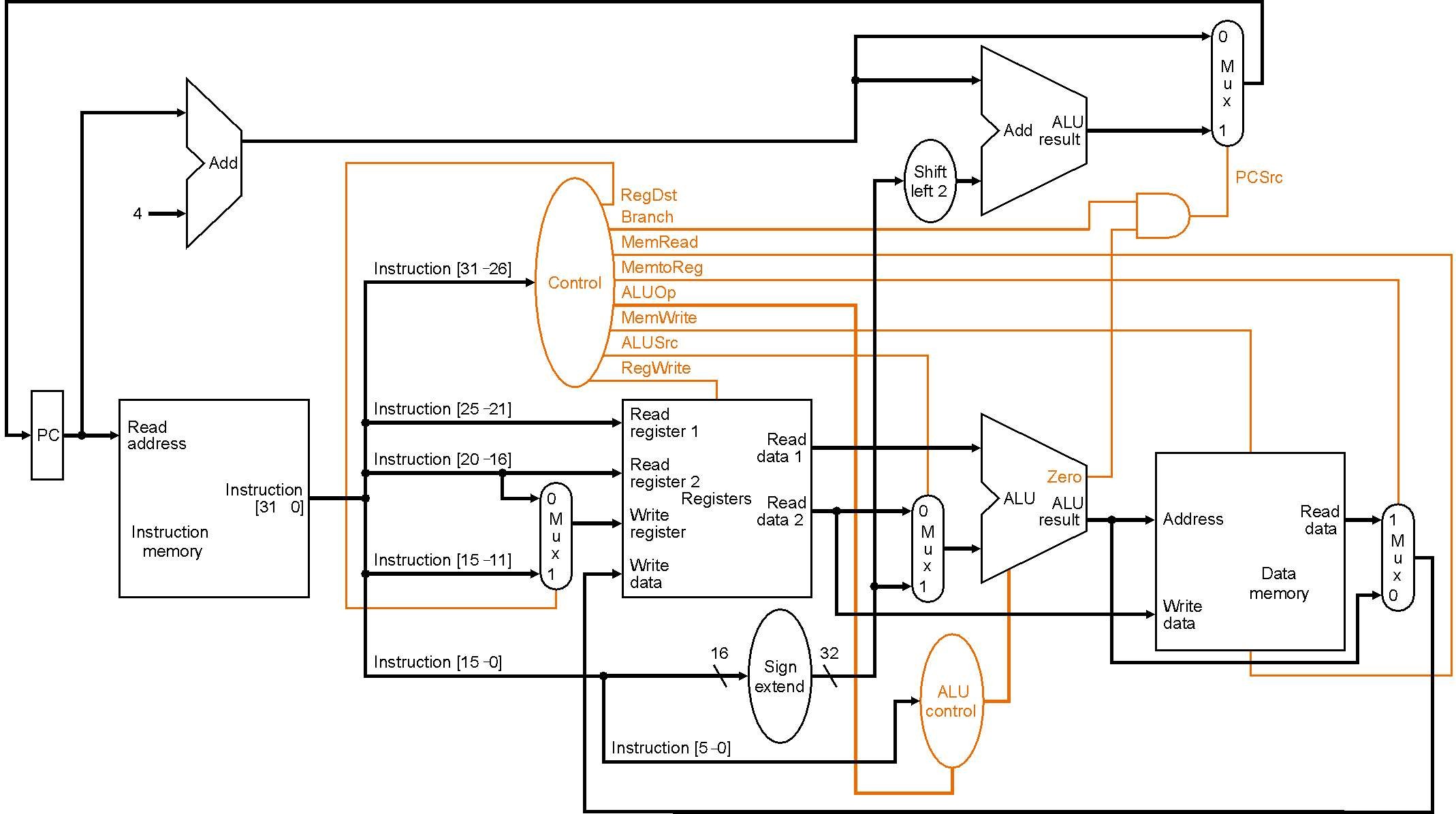


***Слайд*** 13

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **RegDst** | **ALUSrc** | **Memto- Reg** | **Reg Write** | **Mem Read** | **Mem Write** | **Branch** | **ALUOp1** | **ALUp0** |
| R-format | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| lw | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| sw |  |  |  |  |  |  |  | 0 | 0 |
| beq |  |  |  |  |  |  |  | 0 | 1 |

# beq управление





***Слайд*** 14

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **RegDst** | **ALUSrc** | **Memto- Reg** | **Reg Write** | **Mem Read** | **Mem Write** | **Branch** | **ALUOp1** | **ALUp0** |
| R-format | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| lw | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| sw | X | 1 | X | 0 | 0 | 1 | 0 | 0 | 0 |
| beq |  |  |  |  |  |  |  | 0 | 1 |

Таблица истинности для сигналов управления



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **R-format** | **lw** | **sw** | **beq** |
| **Opcode** | | 000000 | 100011 | 101011 | 000100 |
| Outputs | RegDst | 1 | 0 | x | x |
| ALUSrc | 0 | 1 | 1 | 0 |
| MemtoReg | 0 | 1 | x | x |
| RegWrite | 1 | 1 | 0 | 0 |
| MemRead | 0 | 1 | 0 | 0 |
| MemWrite | 0 | 0 | 1 | 0 |
| Branch | 0 | 0 | 0 | 1 |
| ALUOp1 | 1 | 0 | 0 | 0 |
| ALUOp0 | 0 | 0 | 0 | 1 |

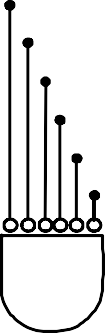
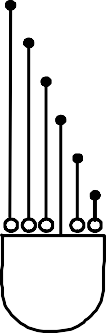
***Слайд*** 15



# Generating the control signals

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **R-** | **format** | **lw** | **sw** | **beq** |
| **Opcode** | | 000000 | | 100011 | 101011 | 000100 |
| Outputs | RegDst | 1 | | 0 | x | x |
| ALUSrc | 0 | | 1 | 1 | 0 |
| MemtoReg | 0 | | 1 | x | x |
| RegWrite | 1 | | 1 | 0 | 0 |
| MemRead | 0 | | 1 | 0 | 0 |
| MemWrite | 0 | | 0 | 1 | 0 |
| Branch | 0 | | 0 | 0 | 1 |
| ALUOp1 | 1 | | 0 | 0 | 0 |
| ALUOp0 | 0 | | 0 | 0 | 1 |

### PLA for control signals



Inputs

Op5

Op4

Op3

Op2

Op1

Op0

Outputs

R-format

Iw

sw

beq

RegDst

ALUSrc

MemtoReg RegWrite MemRead MemWrite Branch

ALUOp1

ALUOpO

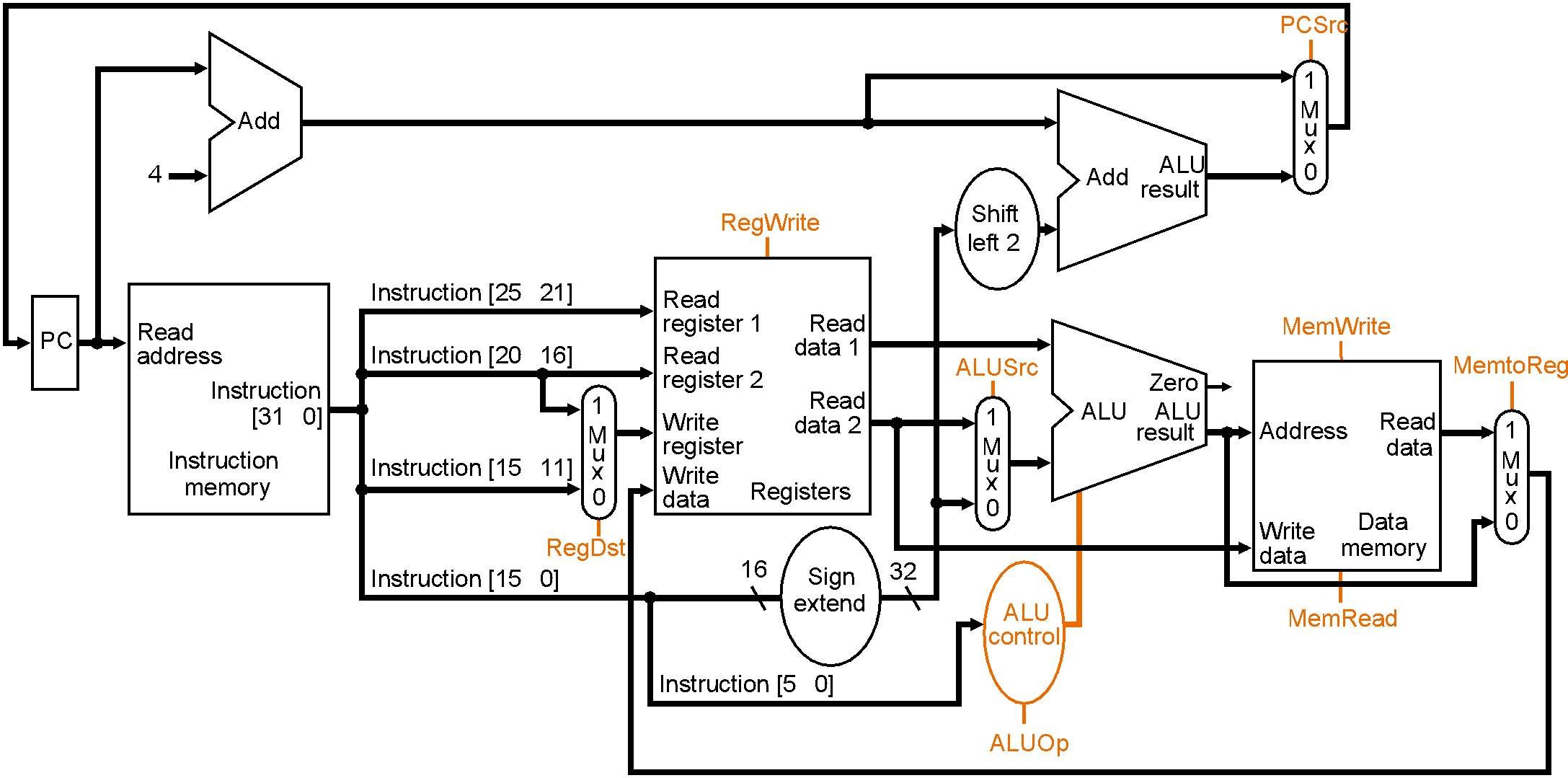
***Слайд*** 16



# Single-Cycle CPU clock cycle time

Критический путь: путь по комбинационной схеме занимающий большее время, чем любой другой.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | I cache | Decode, R-Read | ALU | PC  update | D cache | R-Write | Total |
| R-type | 1 | 1 | .9 | - | - | .8 | 3.7 |
| Load | 1 | 1 | .9 | - | 1 | .8 | 4.7 |
| Store | 1 | 1 | .9 | - | 1 | - | 3.9 |
| beq | 1 | 1 | .9 | .1 | - | - | 3.0 |

Clock cycle time

= 4.7 + setup + hold

***Слайд*** 17



# Однотактный ЦП, итоги

* + Простой
  + Какая команда выполняется дольше всех. Почему это может являться проблемой?
  + Execution time = insts \* cpi \* cycle time
  + Реальные машины имеют намного большее количество команд чем это маленькое подмножество.

***Слайд*** 18

## Почему используется многотактовая схема?



* Проблема: В однотактовой схеме длительность такта должна быть достаточной для выполнения наиболее медленной команды
* Решение: разбиваем выполнение на маленькие части
  + каждая часть выполняется за 1 такт;
  + различные команды потребуют различного количества тактов
* Другое преимущество: Можно использовать меньшее количество логических блоков
  + Одно АЛУ вместо 1 АЛУ и двух сумматоров
  + Можно использовать одну универсальную кэш- память (команды + данные)

***Слайд*** 19



# Многотактовая реализация

Цель: сбалансировать каждую часть работы по времени

3.0

-

-

.1

.9

1

1

beq

3.9

-

.9

Store

4.7

.8

1

-

.9

1

1

Load

3.7

.8

.9

R-type

Total

R-Write

D cache

PC

update

ALU

Decode, R-Read

I cache

-

1

1

1

-

1

1

-

* Load требует 5 тактов
* Store and R-type - 4
* beq - 3

***Слайд*** 20

## Будет ли многотактовая схема быстрее?

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | I  cache | Decode, R-read | ALU | PC  update | D cache | R-write | Total |
| R-type | 1 | 1 | .9 | - | - | .8 | 3.7 |
| Load | 1 | 1 | .9 | - | 1 | .8 | 4.7 |
| Store | 1 | 1 | .9 | - | 1 | - | 3.9 |
| beq | 1 | 1 | .9 | .1 | - | - | 3.0 |

Допустим время установки + удержания = 0.1 ns Однотактовый подход:



Clock cycle time = 4.7 + 0.1 = 4.8 ns

time/inst = 1 cycle/inst \* 4.8 ns/cycle = 4.8 ns/inst Многотактовый design:

Clock cycle time = 1.0 + 0.1 = 1.1 time/inst = CPI \* 1.1 ns/cycle

***Слайд*** 21

## Будет ли многотактовая схема быстрее?

|  |  |  |
| --- | --- | --- |
|  | Cycles needed | Instruction frequency |
| R-type | 4 | 60% |
| Load | 5 | 20% |
| Store | 4 | 10% |
| beq | 3 | 10% |

Какой CPI будет

в данном случае???

Допустим время установки + удержания = 0.1 ns Однотактовый подход:



Clock cycle time = 4.7 + 0.1 = 4.8 ns

time/inst = 1 cycle/inst \* 4.8 ns/cycle = 4.8 ns/inst Многотактовый design:

Clock cycle time = 1.0 + 0.1 = 1.1 time/inst = CPI \* 1.1 ns/cycle

***Слайд*** 22



# The Five Cycles

* Five execution steps (some instructions use fewer)
* IF: Instruction Fetch
* ID: Instruction Decode (& register fetch & add PC+immed)
* EX: Execute
* Mem: Memory access
* WB: Write-Back into registers

IF ID EX Mem WB

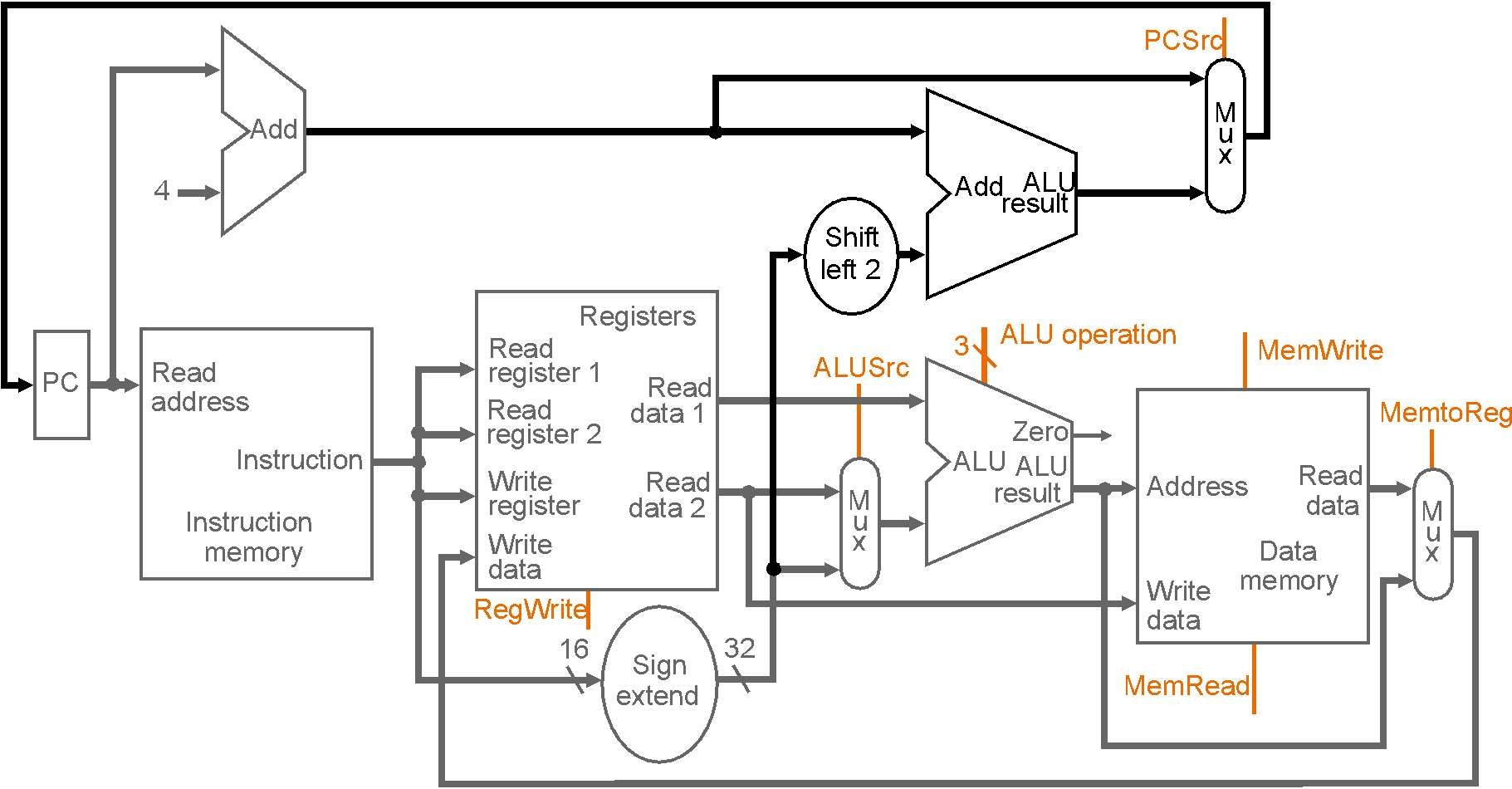
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | I cache | Decode, R-Read | ALU | PC  update | D cache | R-Write | Total |
| R-type | 1 | 1 | .9 | - | - | .8 | 3.7 |
| Load | 1 | 1 | .9 | - | 1 | .8 | 4.7 |
| Store | 1 | 1 | .9 | - | 1 | - | 3.9 |
| beq | 1 | 1 | .9 | .1 | - | - | 3.0 |

***Слайд*** 23



# Partitioning the Single-Cycle Design

IF ID Ex Mem WB



***Слайд*** 24



# Computer of the day

* The British LEO I *(LyonsElectronicOfficeI)*computer, запустил своё первое коммерч. приложение в 1951.
  + Первый компьютер использовавшийся для задач коммерции.
  + 500 kHZ, Multiple input/output buffers
* Построен J. Lyons and Co., a catering and food manufacturing companies (в сотрудничестве с Cambrid University)

ge

* + Just so that their catering service and payroll became efficient!
  + Первые измерения производительности (benchmarks): Bakery Evaluations
  + Первые задачи
    - Отслеживание ежедневных заказов от магазинов, поступающих по телефону каждый день после полудня

The LEO project стал пионером в BPO

Изначально предназначался только для внутреннего использования, но ВМ оказалась столь удачной, что данная чайная компания начала строить ВМ для различных организаций UK

В 1956 Lyons начал обслуживать (payroll calculations) Ford UK и другие компании на LEO I.

**Sowed the seeds for the great computer outsourcing debates *Слайд*** 25



# Вопросы к лекции

Какие биты команды влияют на управление АЛУ?

* Для чего служит поле команды «function»?
* Какие преимущества многотактовых ЦП относительно однотактового?
* Каким образом производить разделение всей схемы ЦП на отдельные стадии?

***Слайд*** 26