

# MC542

## Organização de Computadores Teoria e Prática

2007

Prof. Paulo Cesar Centoducatte

[ducatte@ic.unicamp.br](mailto:ducatte@ic.unicamp.br)

[www.ic.unicamp.br/~ducatte](http://www.ic.unicamp.br/~ducatte)

**MC542**

# **Arquitectura de Computadores**

## **Micro-Arquitectura**

**"DDCA" - (Capítulo 7)**

**"COD" - (Capítulo)**

# Micro-Arquitetura

- **Micro-Arquitetura**
  - **Introdução**
  - **Recursos**
  - **MIPS Mono-Ciclo**
    - » **Lw**
    - » **Sw**
    - » **R-Type**
    - » **Beq**
    - » **j**
  - **MIPS Multi-Ciclos**
    - » **Lw**
    - » **Sw**
    - » **R-Type**
    - » **Beq**
    - » **j**

# Introdução

- **Micro-arquitetura: como está implementada a arquitetura em hardware**
- **Processador:**
  - Datapath: blocos funcionais
  - Controle: sinais de controle

Application Software	programs
Operating Systems	device drivers
Architecture	instructions registers
Micro-architecture	datapaths controllers
Logic	adders memories
Digital Circuits	AND gates NOT gates
Analog Circuits	amplifiers filters
Devices	transistors diodes
Physics	electrons

# Micro-Arquitetura

- **Múltiplas implementações para uma mesma arquitetura:**
  - **Single-cycle**
    - » Cada instrução é executada em um único ciclo
  - **Multicycle**
    - » A execução de cada instrução é dividida em uma série de passos menores
  - **Pipelined**
    - » A execução de cada instrução é dividida em uma série de passos menores
    - » Múltiplas instruções (parte de) executando ao mesmo tempo.

# Micro-Arquitetura

- Program execution time

$$\text{Execution Time} = (\# \text{ instructions})(\text{cycles/instruction})(\text{seconds/cycle})$$

- Definições:

- Cycles/instruction = CPI
- Seconds/cycle = clock period
- $1/\text{CPI} = \text{Instructions/cycle} = \text{IPC}$

- Desafios na implementação de uma micro-arquitetura
  - Custo
  - Power
  - Desempenho

# Micro-Arquitetura

- Processador MIPS (**M**icroprocessor without **I**nterlocked **P**ipeline **S**tages)
  - Vamos implementar um sub conjunto das instruções MIPS:
  - R-type instructions: and, or, add, sub, slt
  - Memory instructions: lw, sw
  - Branch instructions: beq

# Micro-Arquitetura

- Estado da Arquitetura

- Determina o estado do Processador em um dado instante de tempo

PC

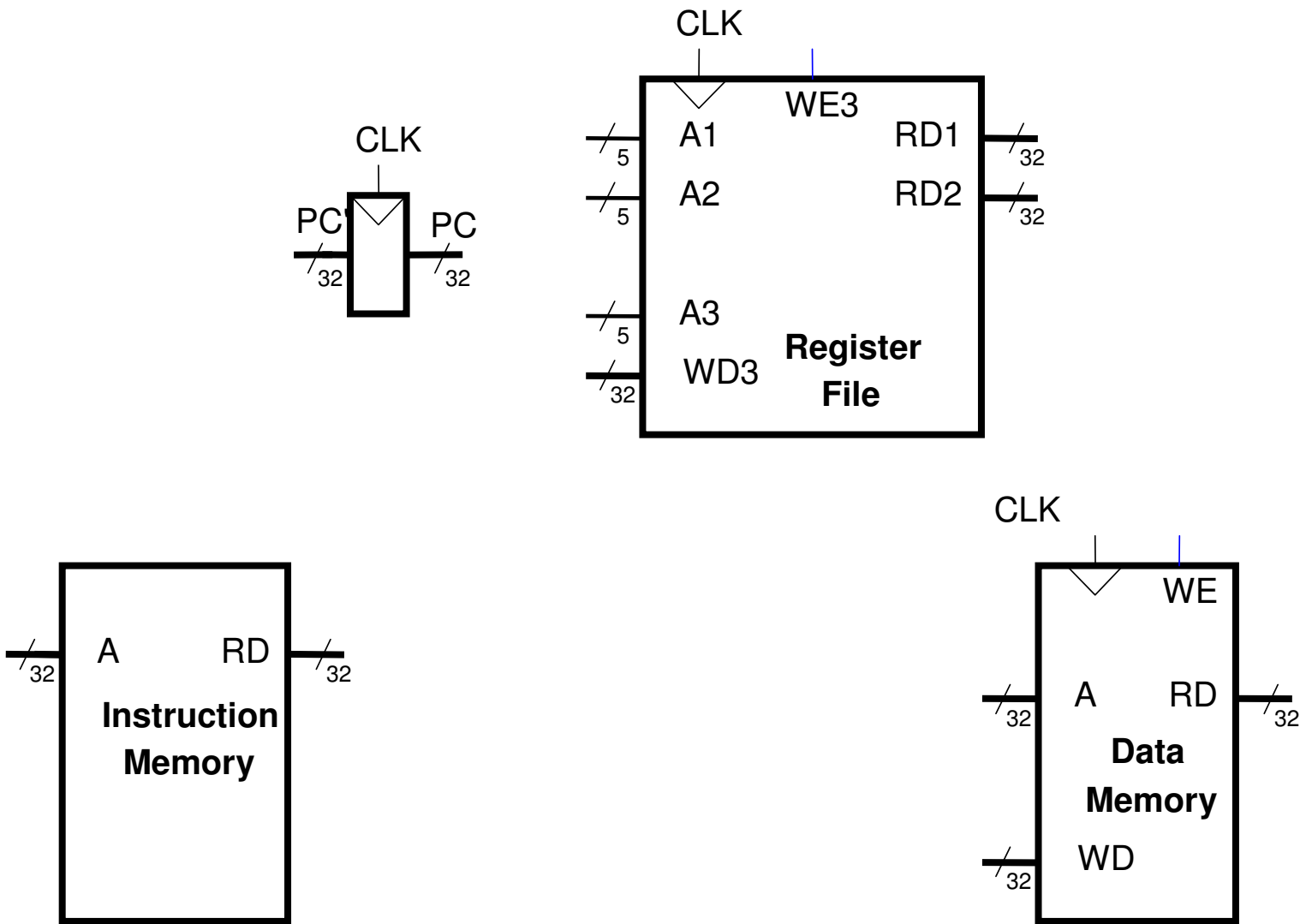
32 registradores

Memória



# Micro-Arquitetura

- Elementos de estados do MIPS:

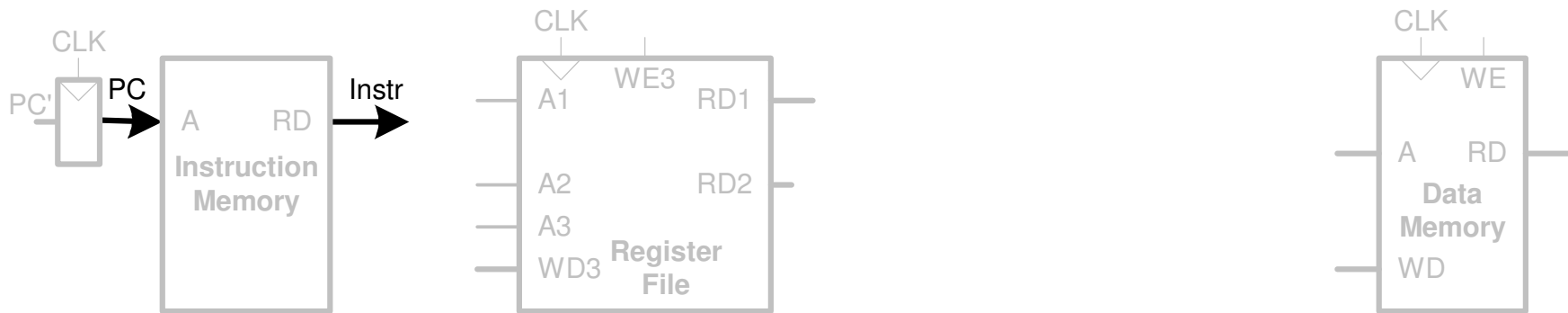


# Processador MIPS Single-Cycle

- Datapath
- Controle

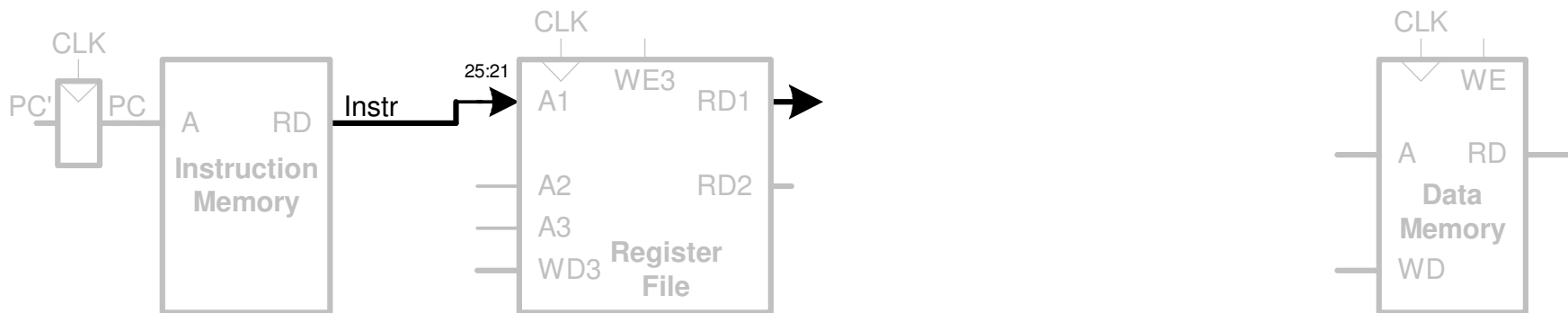
# Processador MIPS Single-Cycle

- Execução de `lw`
  - 1: Fetch da Instrução



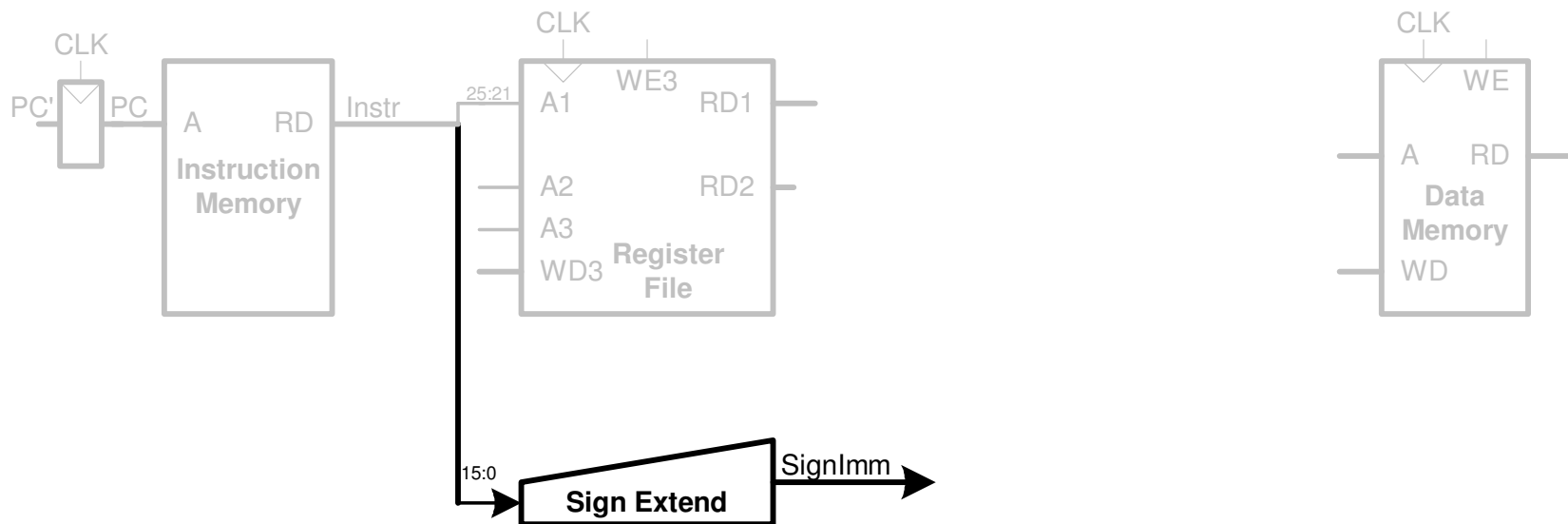
# Processador MIPS Single-Cycle

- Execução de `lw`  
2: Lê o operando fonte do RF



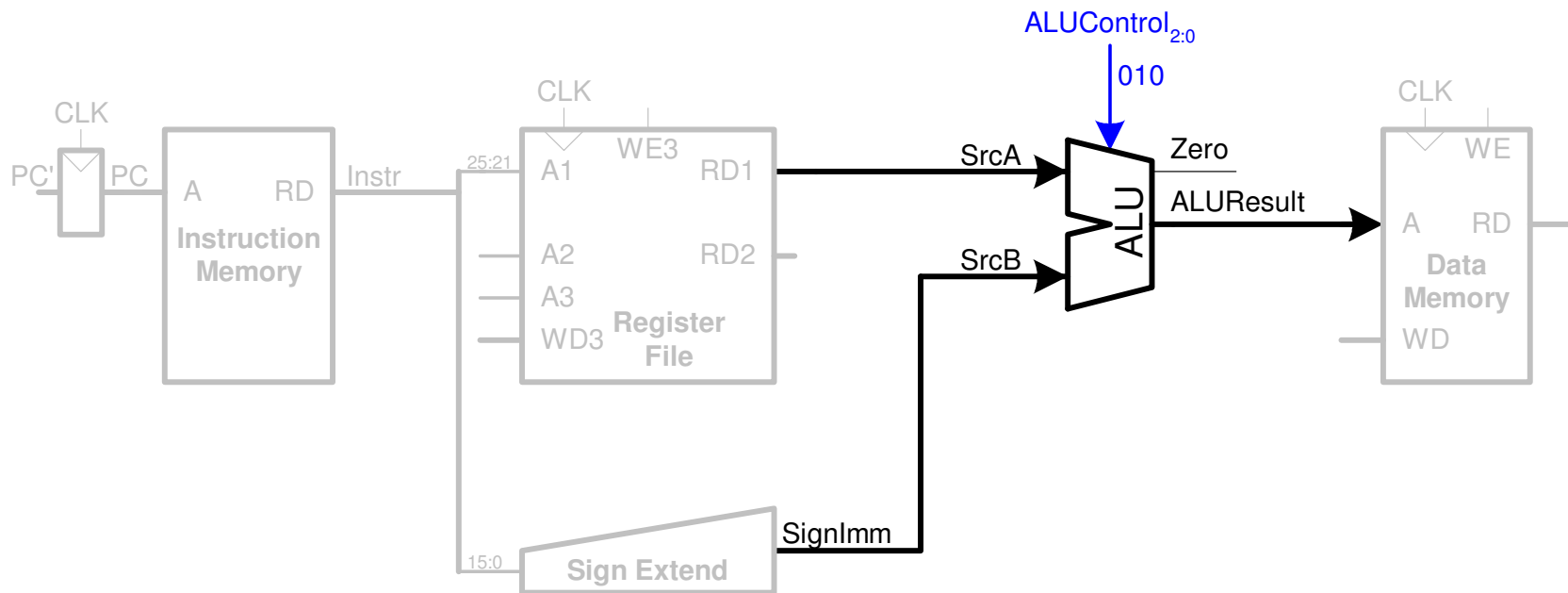
# Processador MIPS Single-Cycle

- Execução de `lw`
  - 3: Sign-extend o imediato



# Processador MIPS Single-Cycle

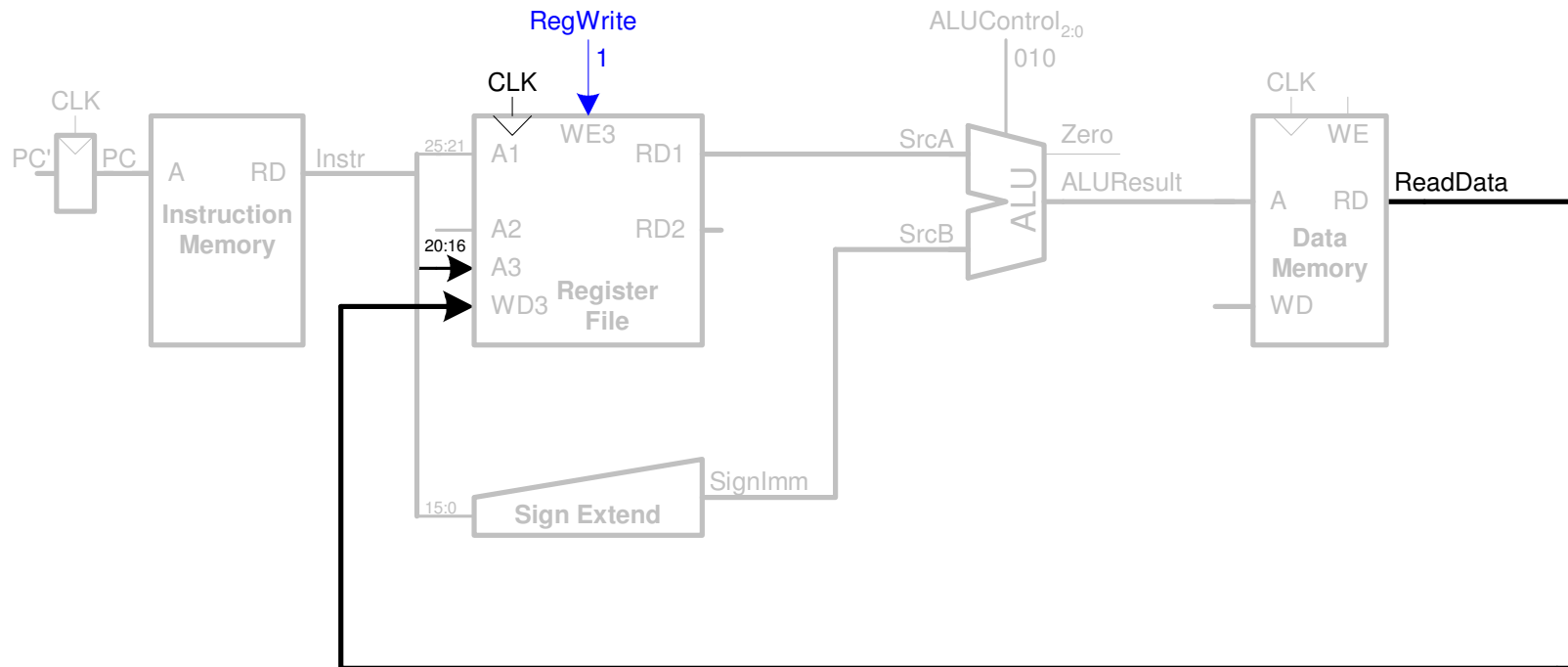
- Execução de `lw`
  - 1: Busca a instrução na memória de instrução
  - 2: Decodifica a instrução
  - 3: Busca os dados na memória de dados
  - 4: Calcula o endereço efetivo de memória



# Processador MIPS Single-Cycle

- Execução de lw

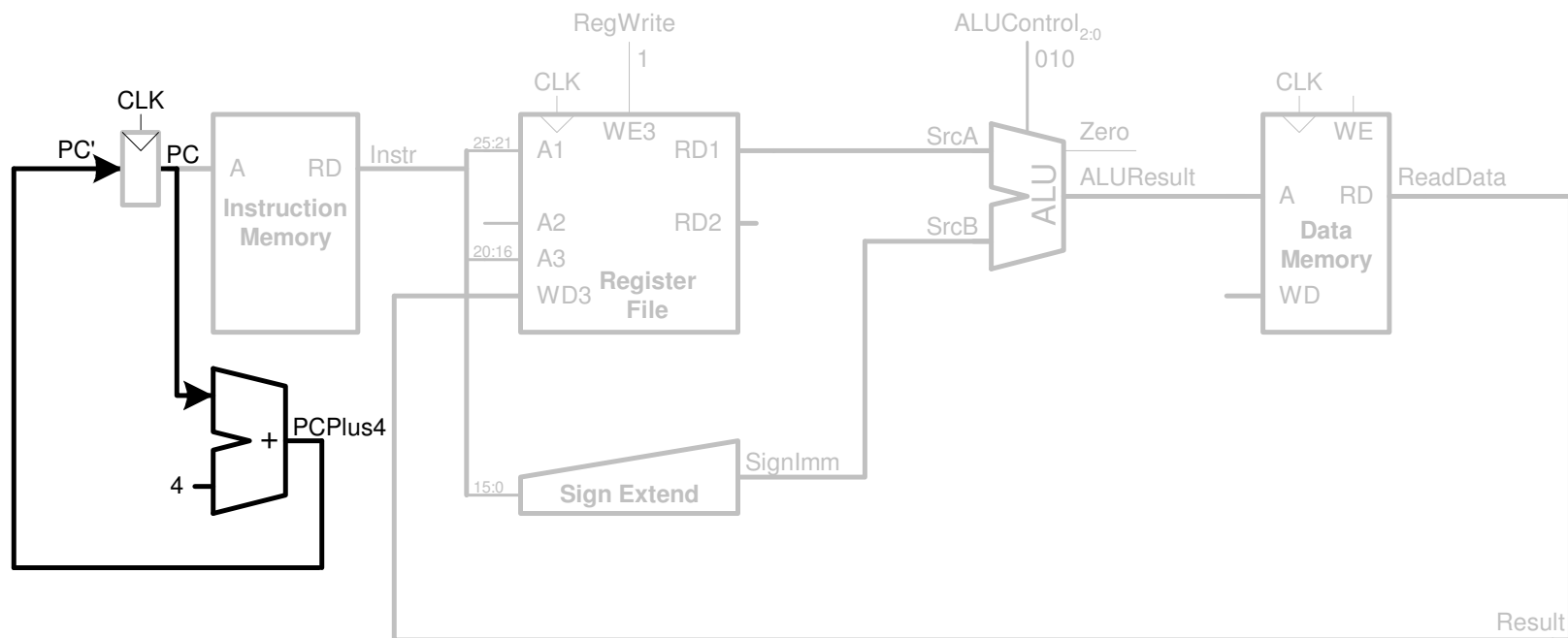
5: Lê o dado da memória e o escreva no RF



# Processador MIPS Single-Cycle

- Execução de `lw`

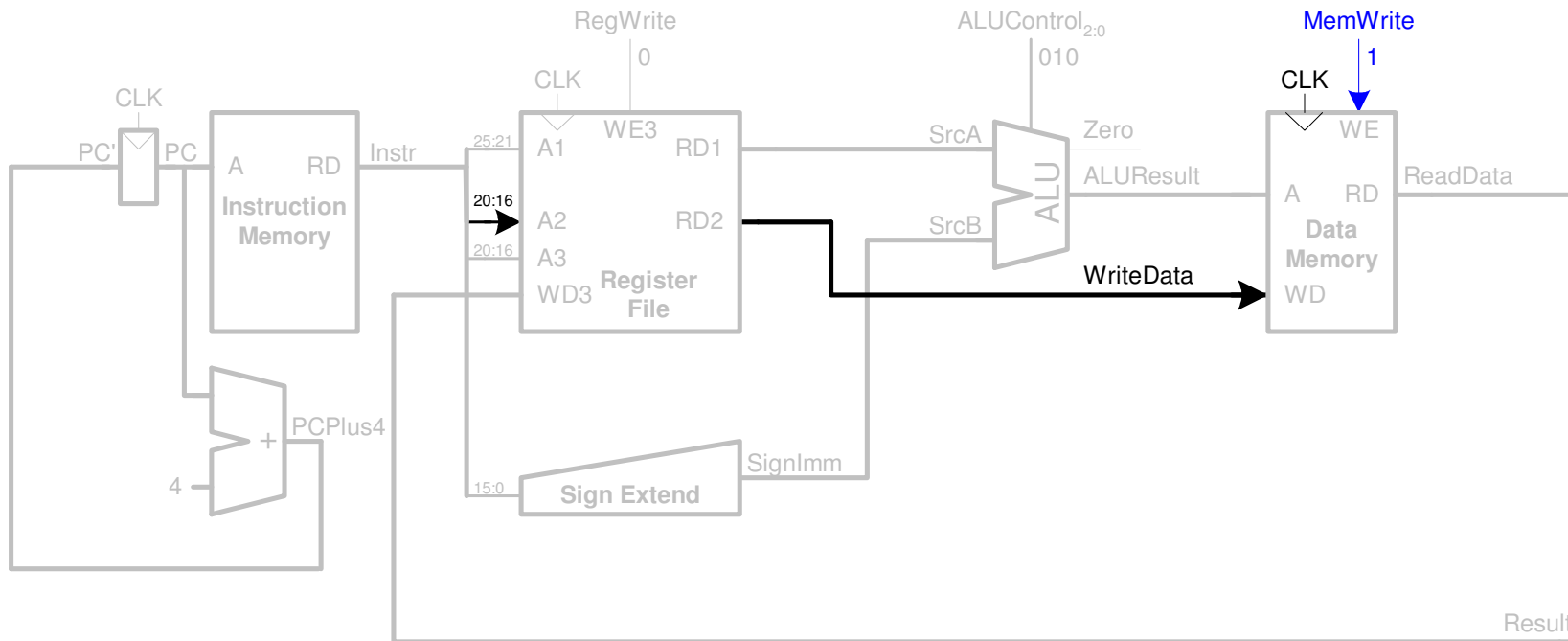
6: Determina o endereço da próxima instrução





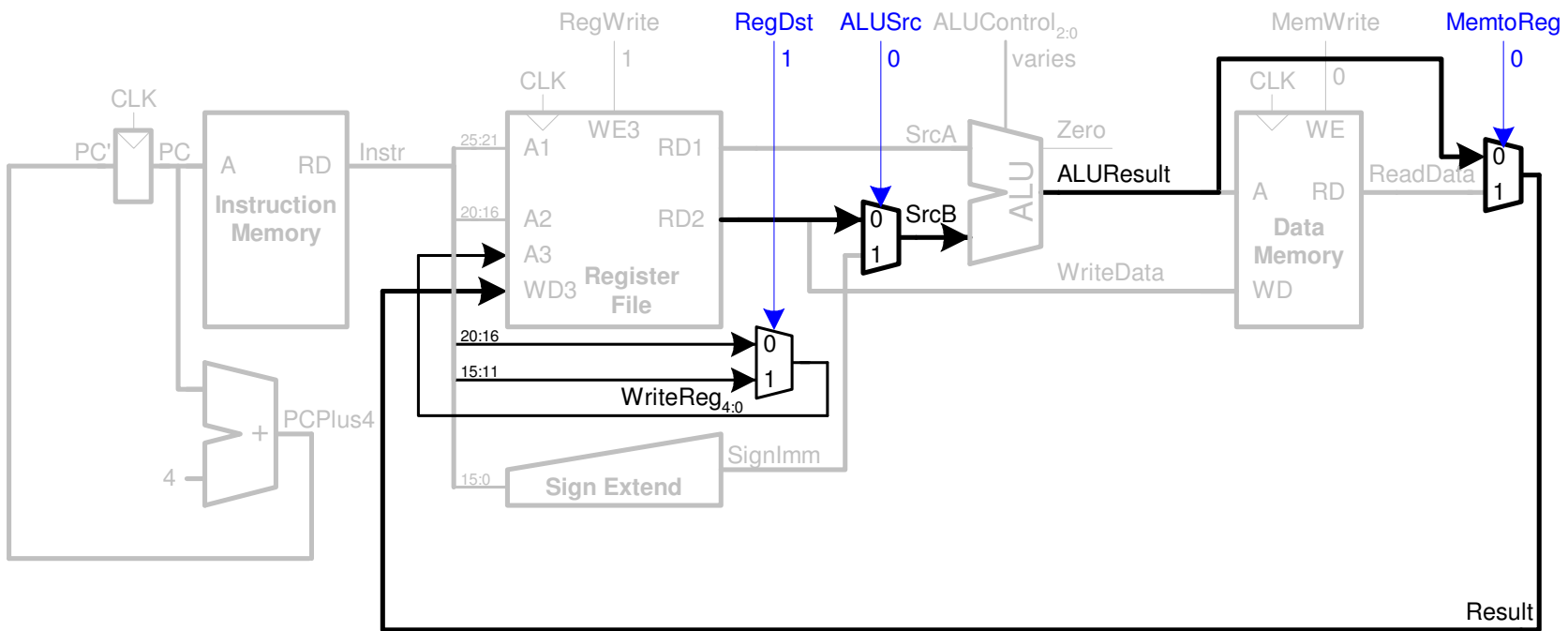
# Processador MIPS Single-Cycle

- Execução de sw
- Precisa escrever o valor do registrador na memória



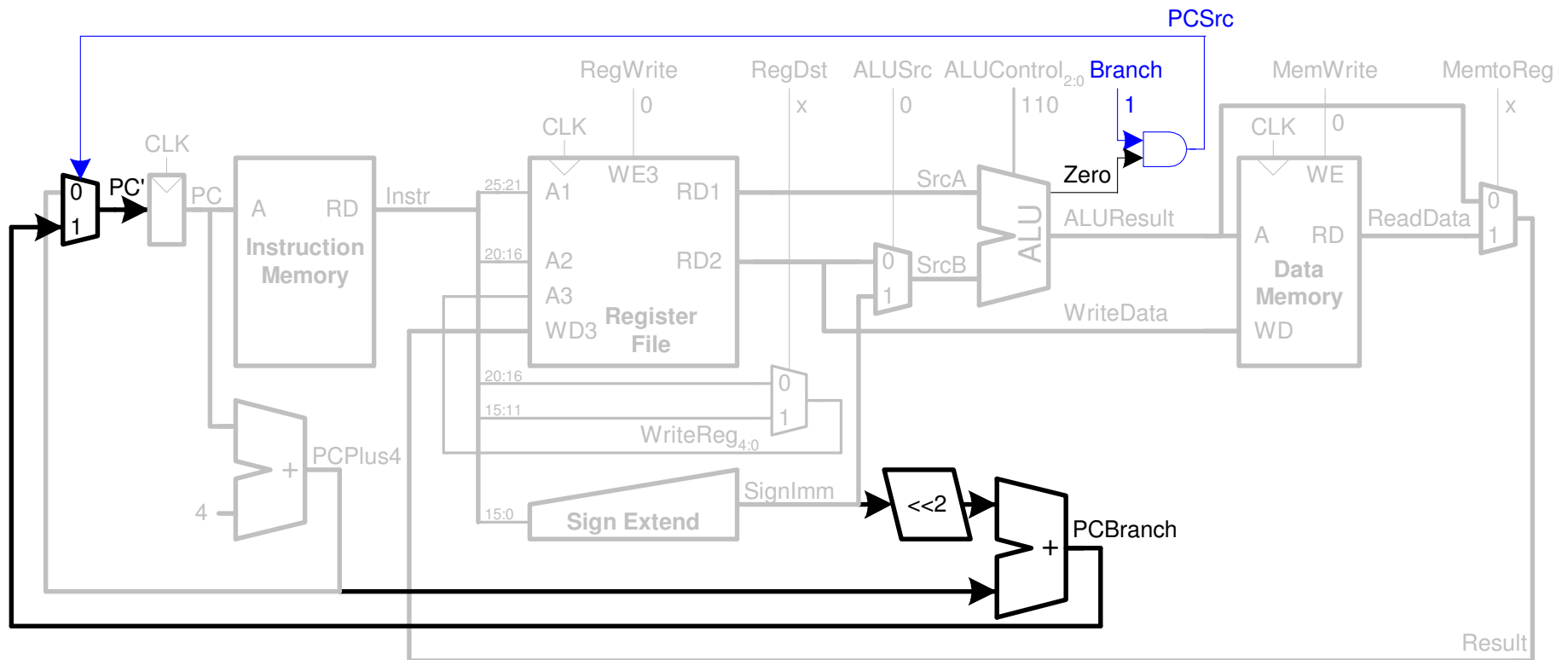
# Processador MIPS Single-Cycle

- Instruções R-Type: add, sub, and, or, ...
- Escrever *ALUResult* no RF
  - Escreve em rd e não em rt

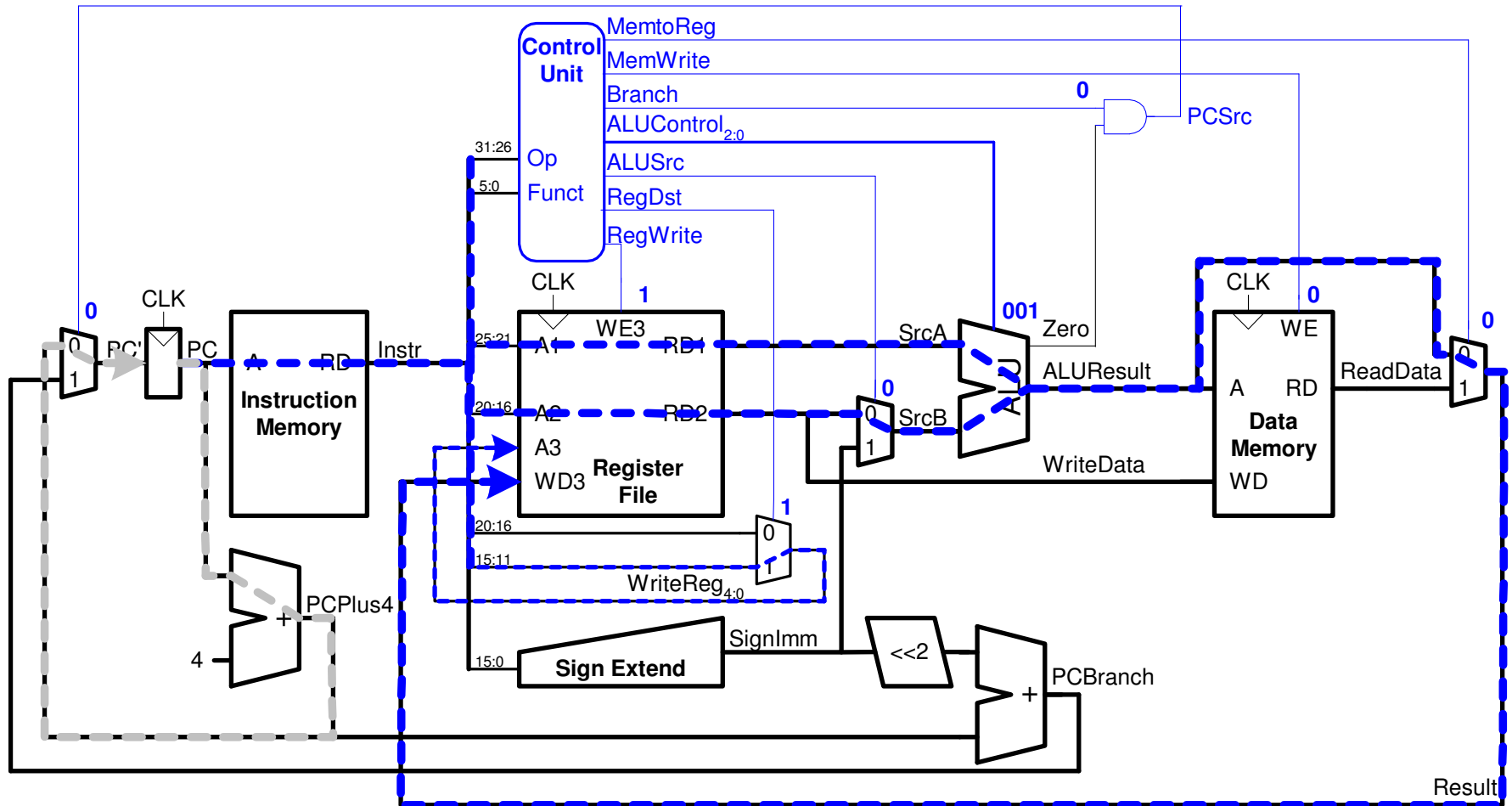


# Processador MIPS Single-Cycle

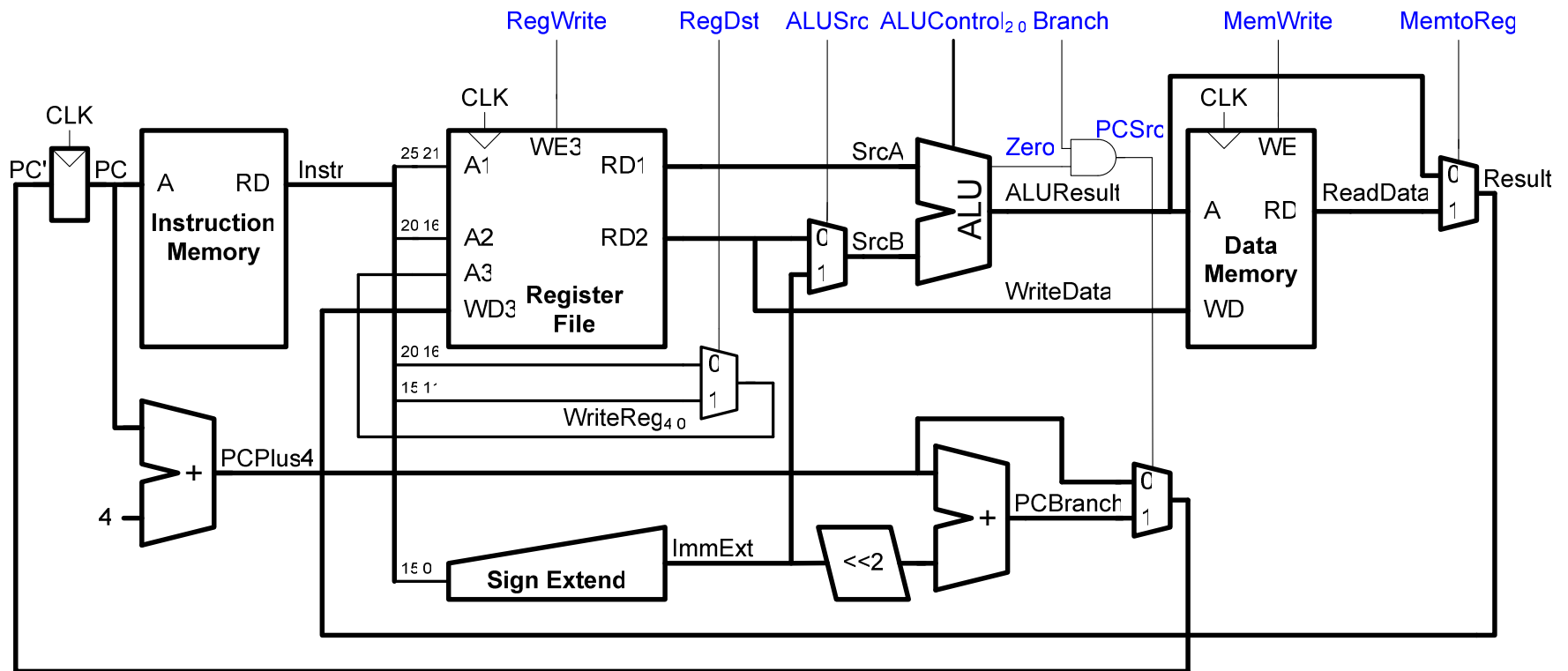
- Instrução beq
- Determina se os conteúdos dos registradores são iguais
- Calcula o endereço alvo do desvio (**sign-extended immediate + PC+4**)



# Processador MIPS Single-Cycle - or

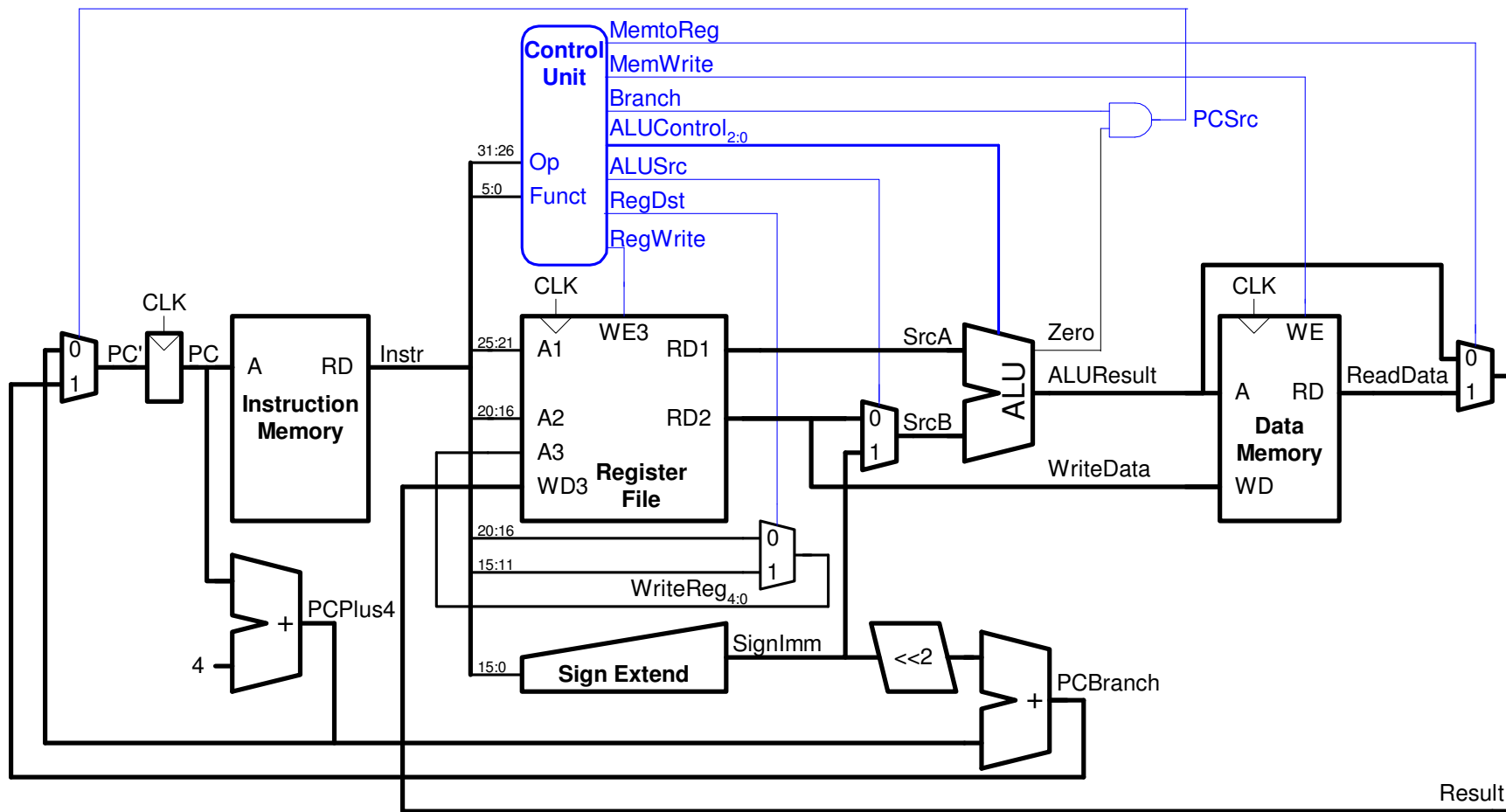


# Processador MIPS Single-Cycle - sw



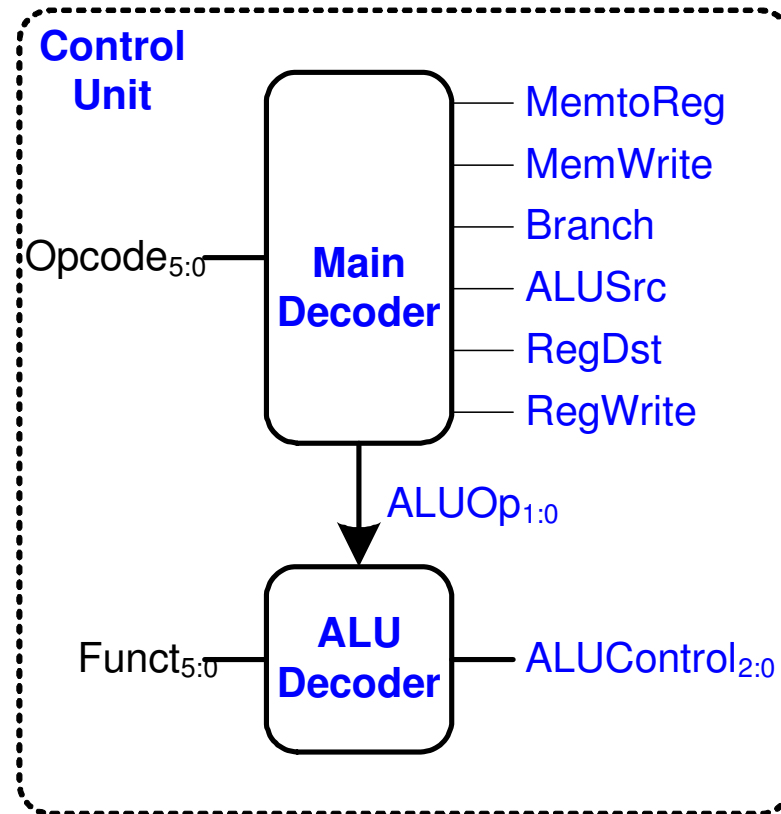
# Processador MIPS Single-Cycle

- Unidade de Controle



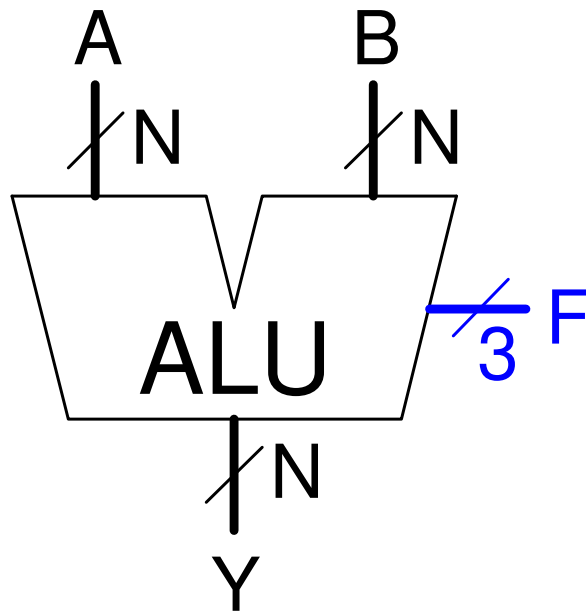
# Processador MIPS Single-Cycle

## Control Unit



# Processador MIPS Single-Cycle

- ALU

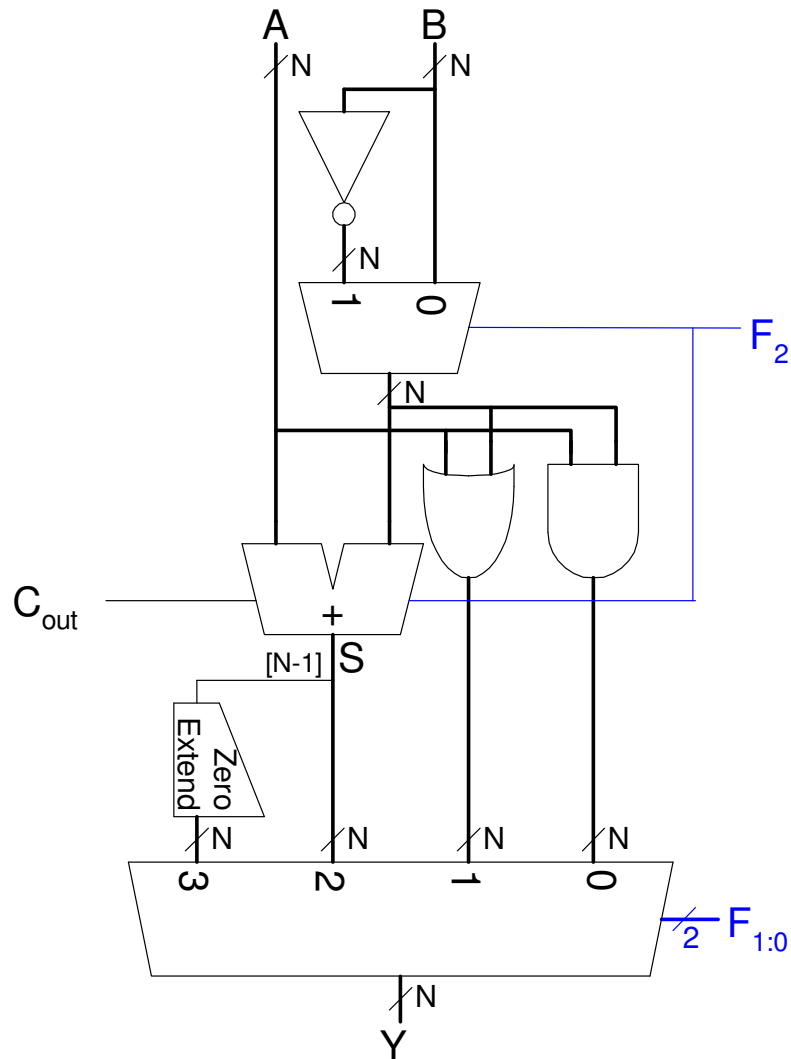


$F_{2:0}$	Function
000	A & B
001	A   B
010	A + B
011	not used
100	A & B
101	A   ~B
110	A - B
111	SLT



# Processador MIPS Single-Cycle

- ALU



# Processador MIPS Single-Cycle

- ALU Decoder

$ALUOp_{1:0}$	Meaning
00	Add
01	Subtract
10	Look at Funct
11	Not Used

$ALUOp_{1:0}$	Funct	$ALUControl_{2:0}$
00	X	010 (Add)
X1	X	110 (Subtract)
1X	100000 (add)	010 (Add)
1X	100010 (sub)	110 (Subtract)
1X	100100 (and)	000 (And)
1X	100101 (or)	001 (Or)
1X	101010 (slt)	111 (SLT)

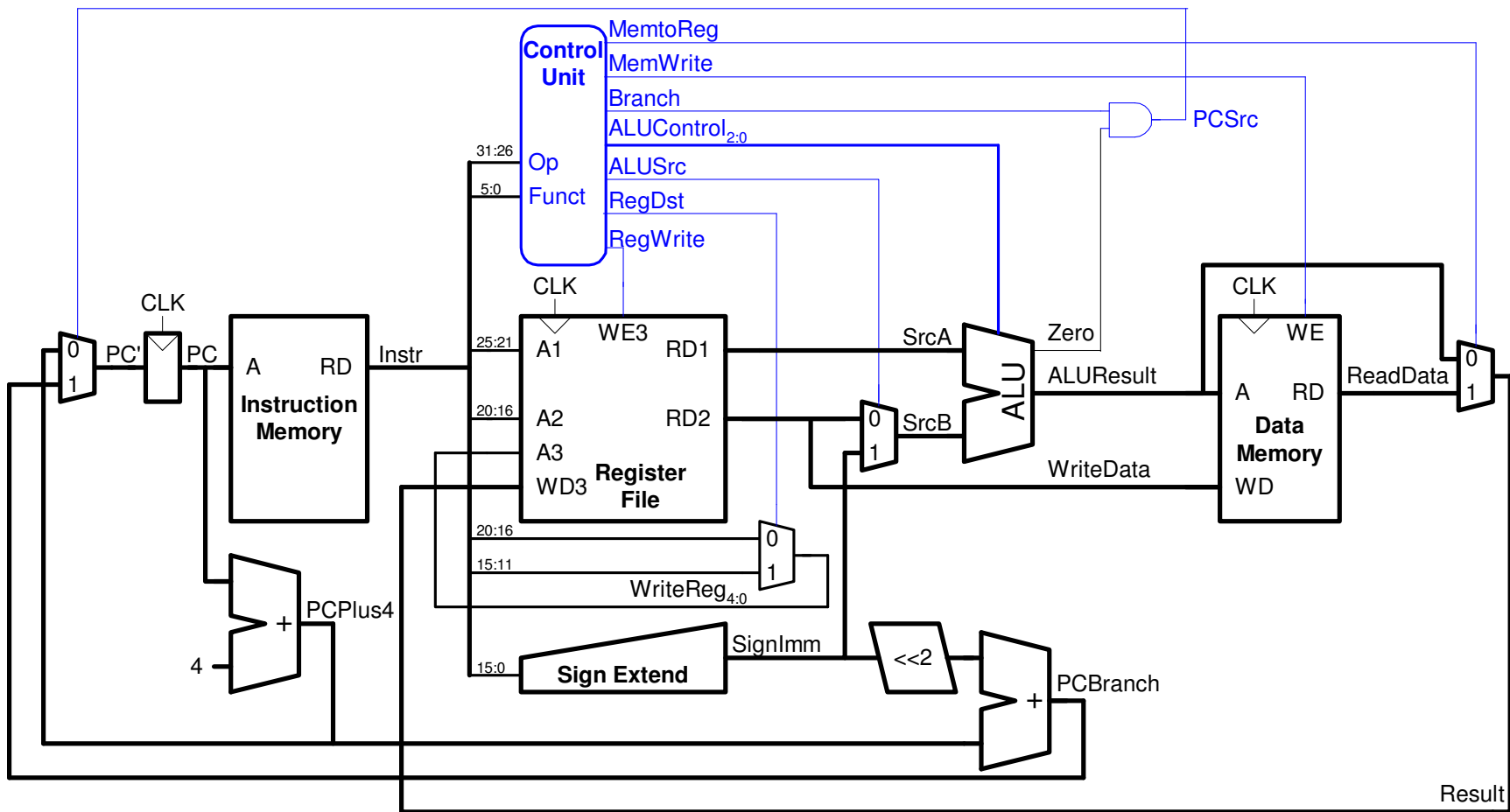
# Processador MIPS Single-Cycle

- Decodificador Principal

Instruction	Op <sub>5:0</sub>	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUOp <sub>1:0</sub>
R-type	000000	1	1	0	0	0	0	10
lw	100011	1	0	1	0	0	0	00
sw	101011	0	X	1	0	1	X	00
beq	000100	0	X	0	1	0	X	01

# Processador MIPS Single-Cycle

- **Extendendo a Funcionalidade: addi**



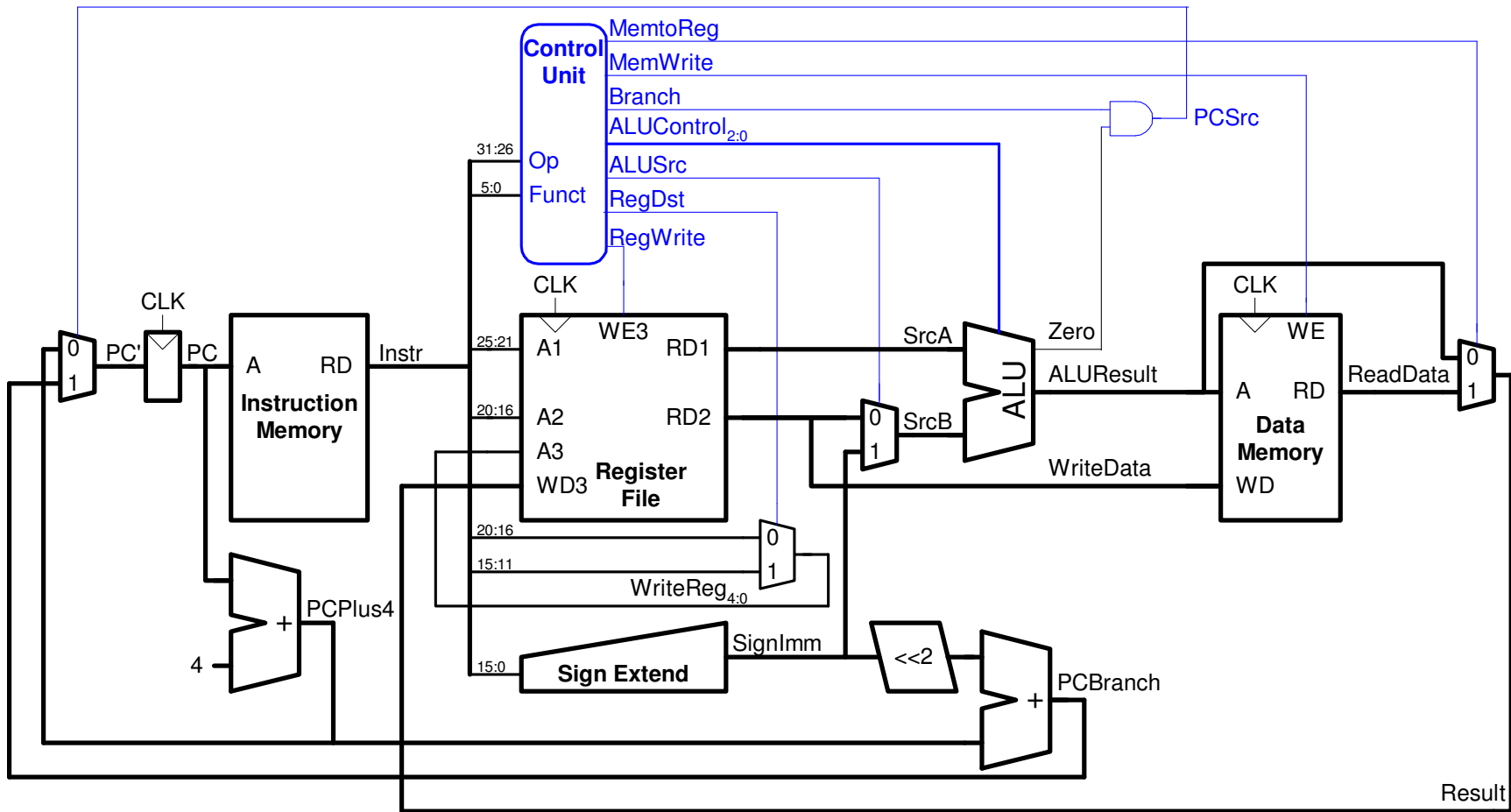
# Processador MIPS Single-Cycle

- Decodificador Principal

Instruction	Op <sub>5:0</sub>	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUOp <sub>1:0</sub>
R-type	000000	1	1	0	0	0	0	10
lw	100011	1	0	1	0	0	1	00
sw	101011	0	X	1	0	1	X	00
beq	000100	0	X	0	1	0	X	01
addi	001000	1	0	1	0	0	0	00

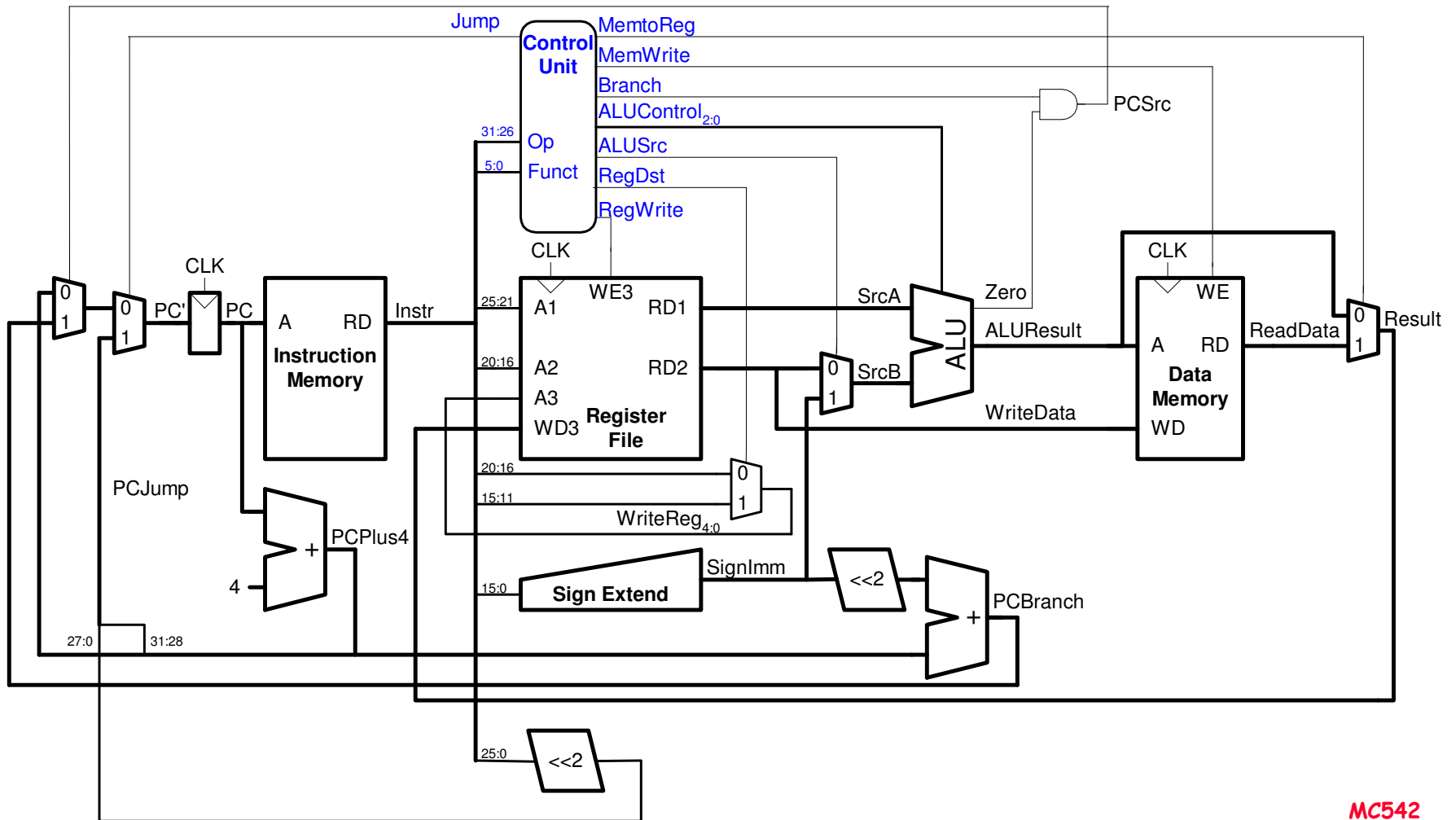
# Processador MIPS Single-Cycle

- **Extendendo a Funcionalidade: j**



# Processador MIPS Single-Cycle

- **Extendendo a Funcionalidade: j**



# Processador MIPS Single-Cycle

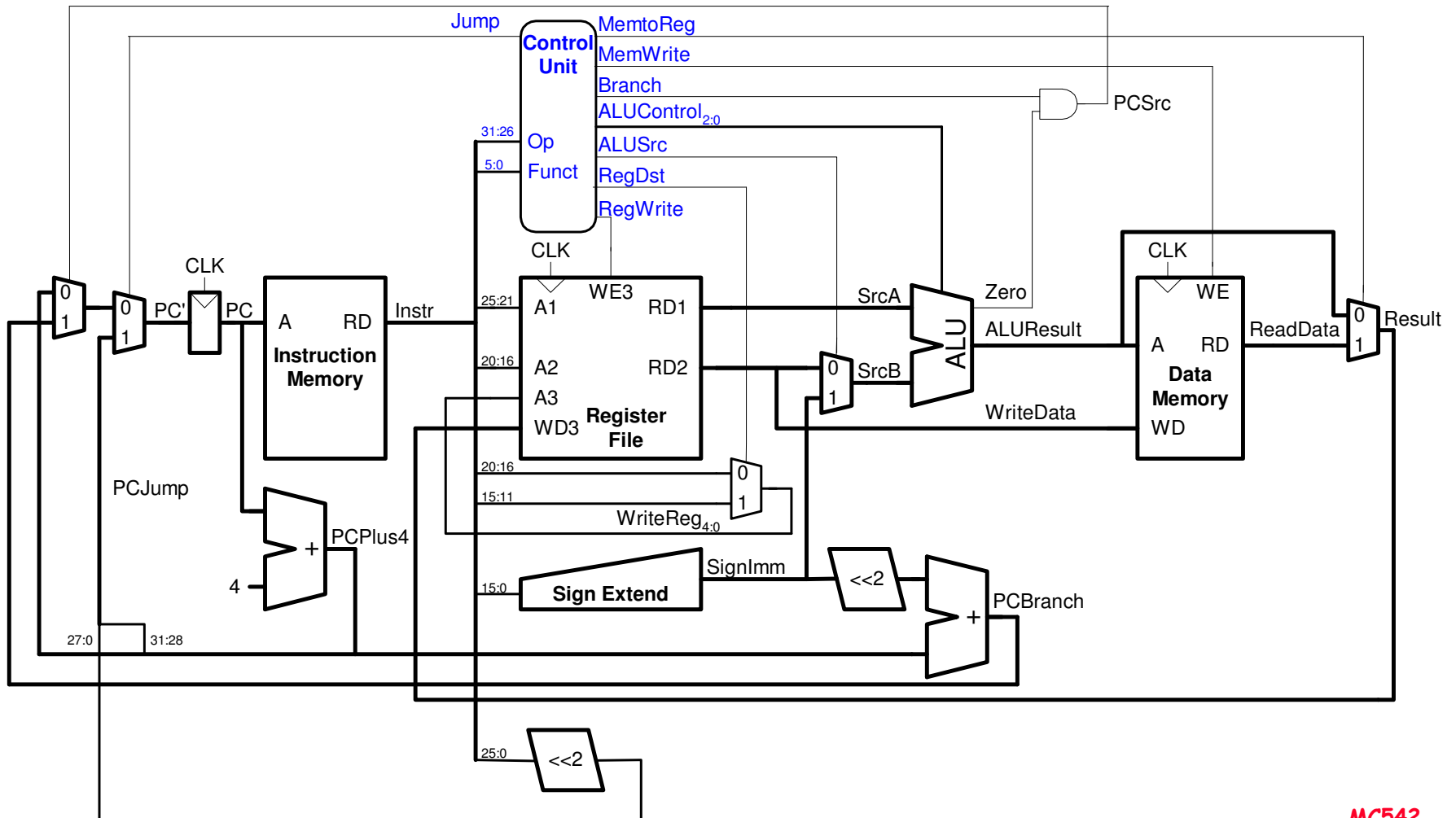
- Decodificador Principal

Instruction	Op <sub>5:0</sub>	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUOp <sub>1:0</sub>	Jump
R-type	000000	1	1	0	0	0	0	10	0
lw	100011	1	0	1	0	0	1	00	0
sw	101011	0	X	1	0	1	X	00	0
beq	000100	0	X	0	1	0	X	01	0
addi	001000	1	0	1	0	0	0	00	0
j	000100	0	X	X	X	0	X	XX	1



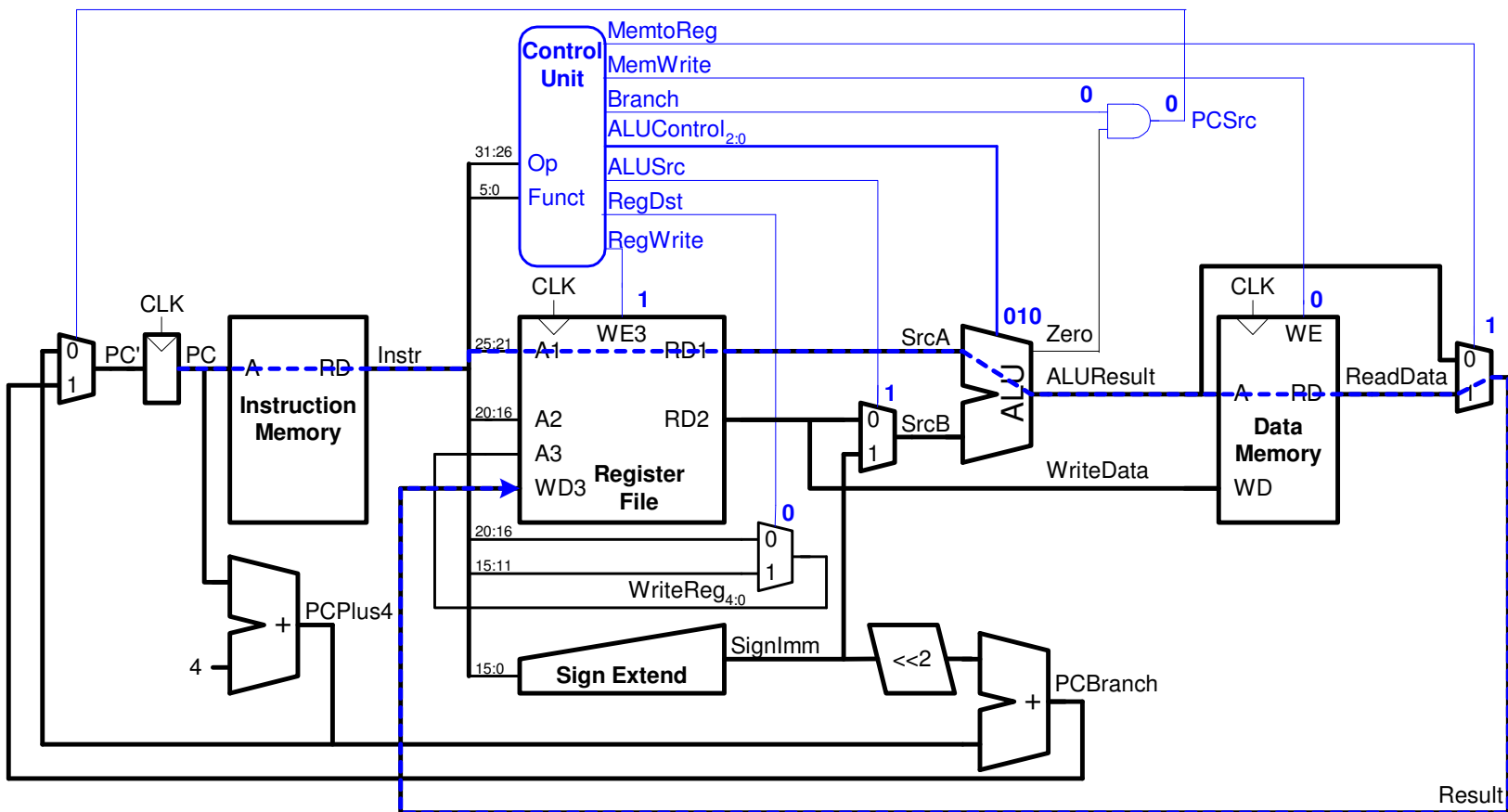
# Processador MIPS Single-Cycle

- Desempenho: Quão rápido é o processador?



# Processador MIPS Single-Cycle

- O cycle time é limitado pelo caminho crítico: 1w



# Processador MIPS Single-Cycle

- Caminho crítico

$$T_c = t_{pcq\_PC} + t_{mem} + \max(t_{RFread}, t_{sext}) + t_{mux} + t_{ALU} + t_{mem} + t_{mux} + t_{RFsetup}$$

- Na maioria das implementações os caminhos limitantes são: memória, ALU, register file. Assim,

$$T_c = t_{pcq\_PC} + 2t_{mem} + t_{RFread} + 2t_{mux} + t_{ALU} + t_{RFsetup}$$

# Processador MIPS Single-Cycle

Element	Parameter	Delay (ps)
Register clock-to-Q	$t_{pcq\_PC}$	30
Register setup	$t_{setup}$	20
Multiplexer	$t_{mux}$	25
ALU	$t_{ALU}$	200
Memory read	$t_{mem}$	250
Register file read	$t_{RFread}$	150
Register file setup	$t_{RFsetup}$	20

$$\begin{aligned}T_c &= t_{pcq\_PC} + 2t_{mem} + t_{RFread} + 2t_{mux} + t_{ALU} + t_{RFsetup} \\ &= [30 + 2(250) + 150 + 2(25) + 200 + 20] \text{ ps} \\ &= 950 \text{ ps}\end{aligned}$$

# Processador MIPS Single-Cycle

- Para um programa 100 bilhões de instruções executando em um processador MIPS single-cycle,

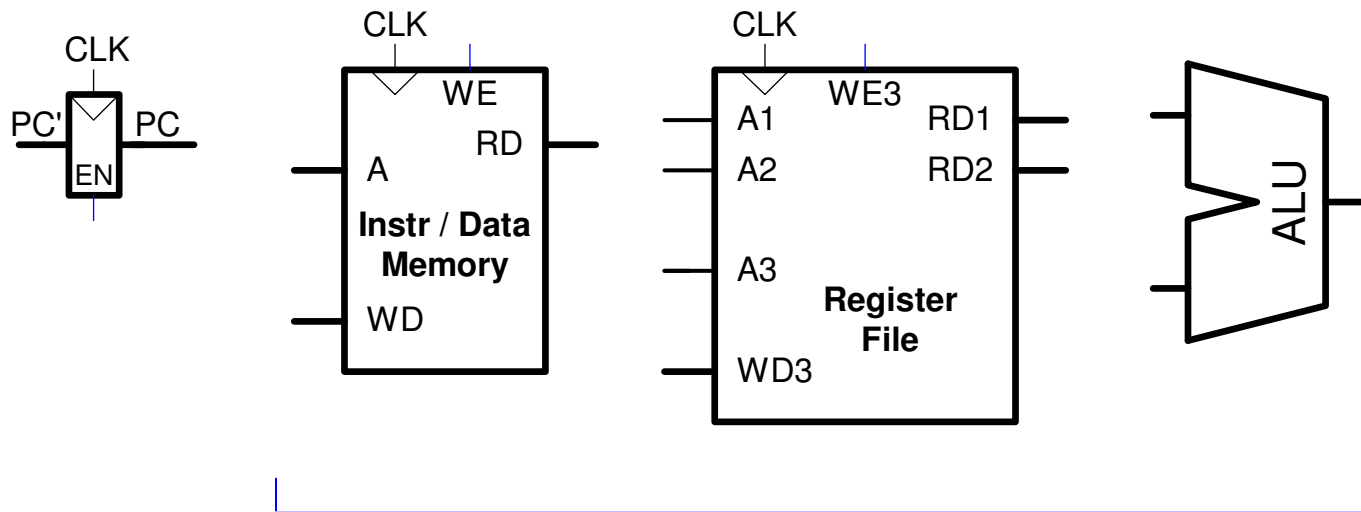
$$\begin{aligned}\text{Execution Time} &= (\# \text{ instructions})(\text{cycles/instruction})(\text{seconds/cycle}) \\ &= (100 \times 10^9)(1)(950 \times 10^{-12} \text{ s}) \\ &= 95 \text{ seconds}\end{aligned}$$

# Processador MIPS Multicycle

- Datapath
- Controle

# Processador MIPS Multicycle

- Blocos Básicos

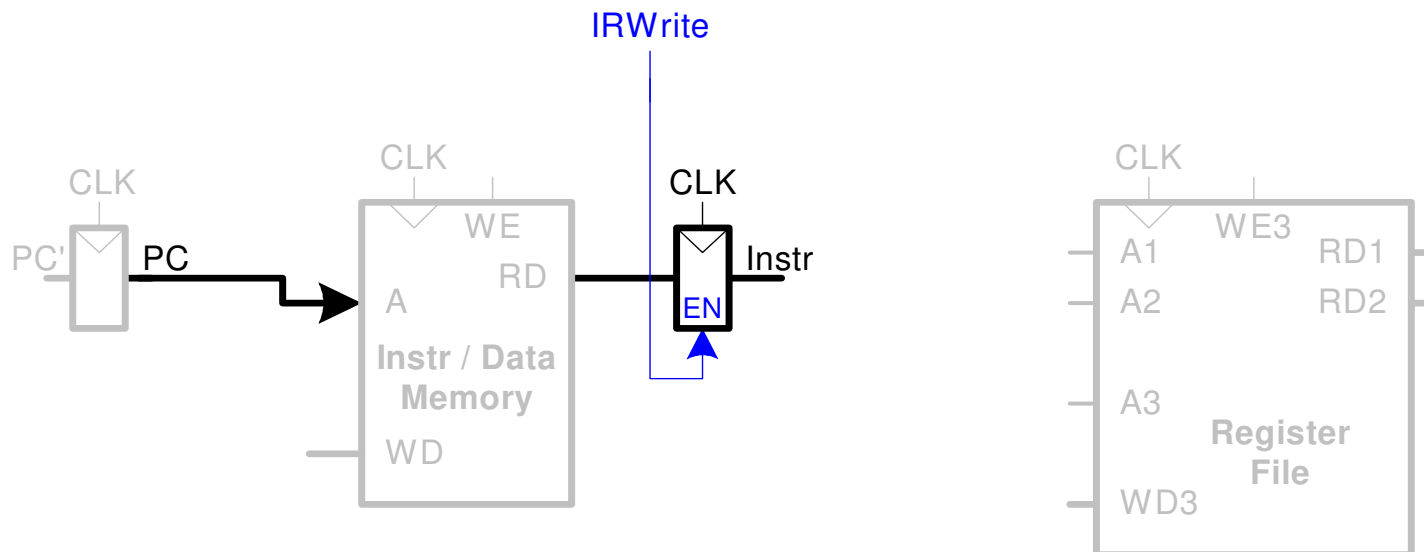


**Restrição de Projeto:**

**Somente 1 pode ser usado por vez**

# Processador MIPS Multicycle

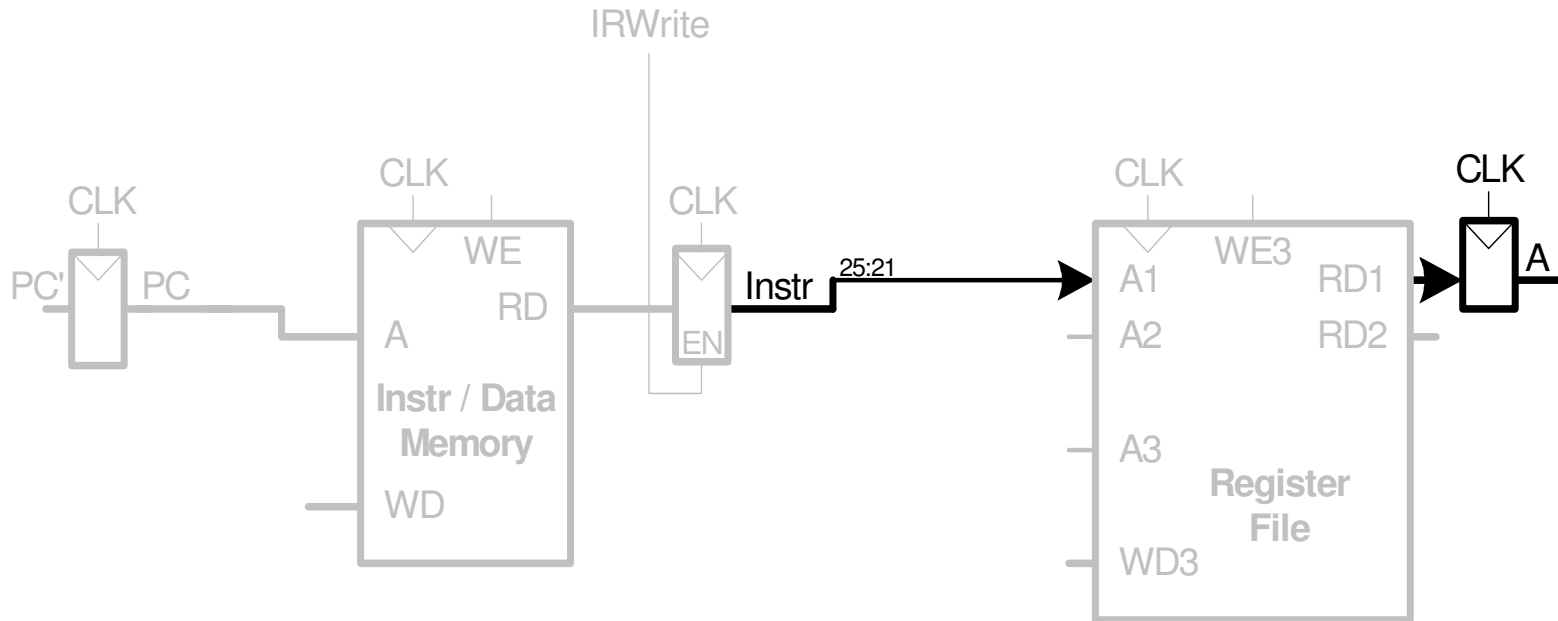
- Datapath
  - datapath para o lw
- **Exemplo:** lw \$s4, 27(\$t2)
- Fetch da Instrução





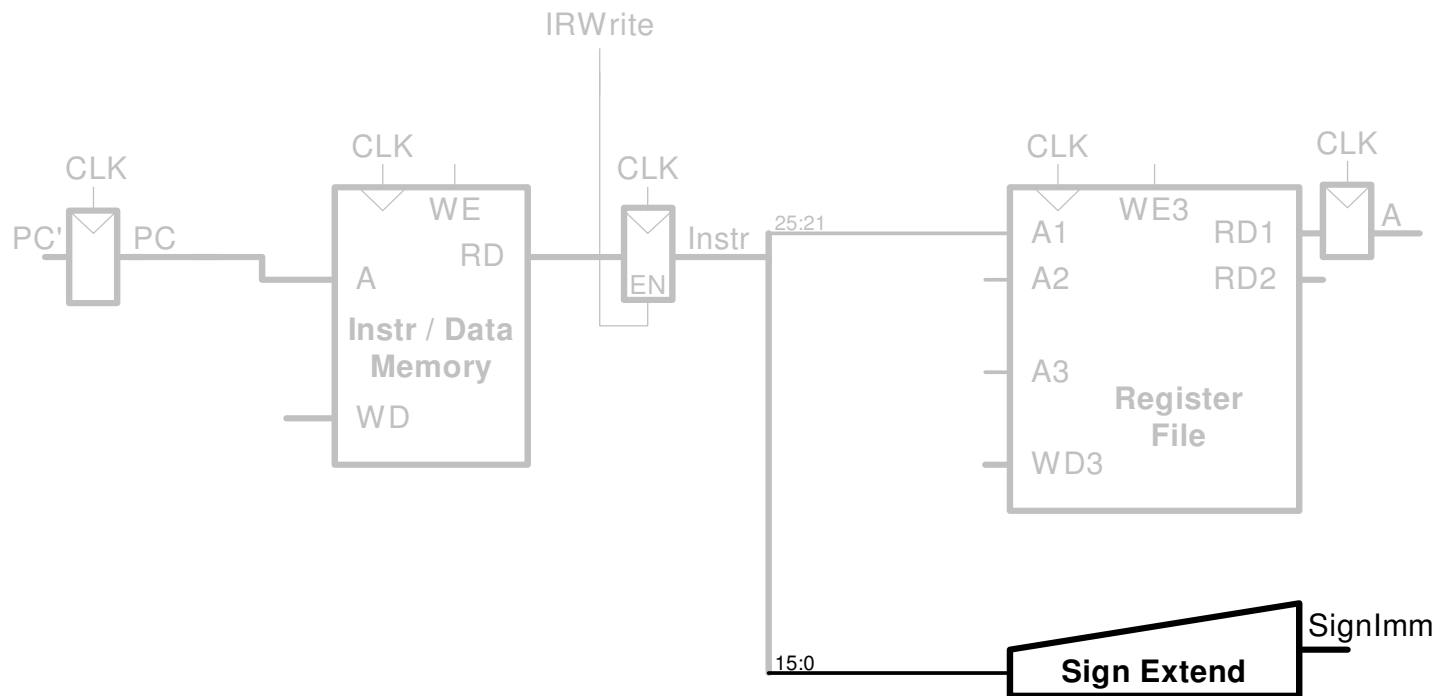
# Processador MIPS Multicycle

- Leitura do operando fonte do RF ( $r_s$ )



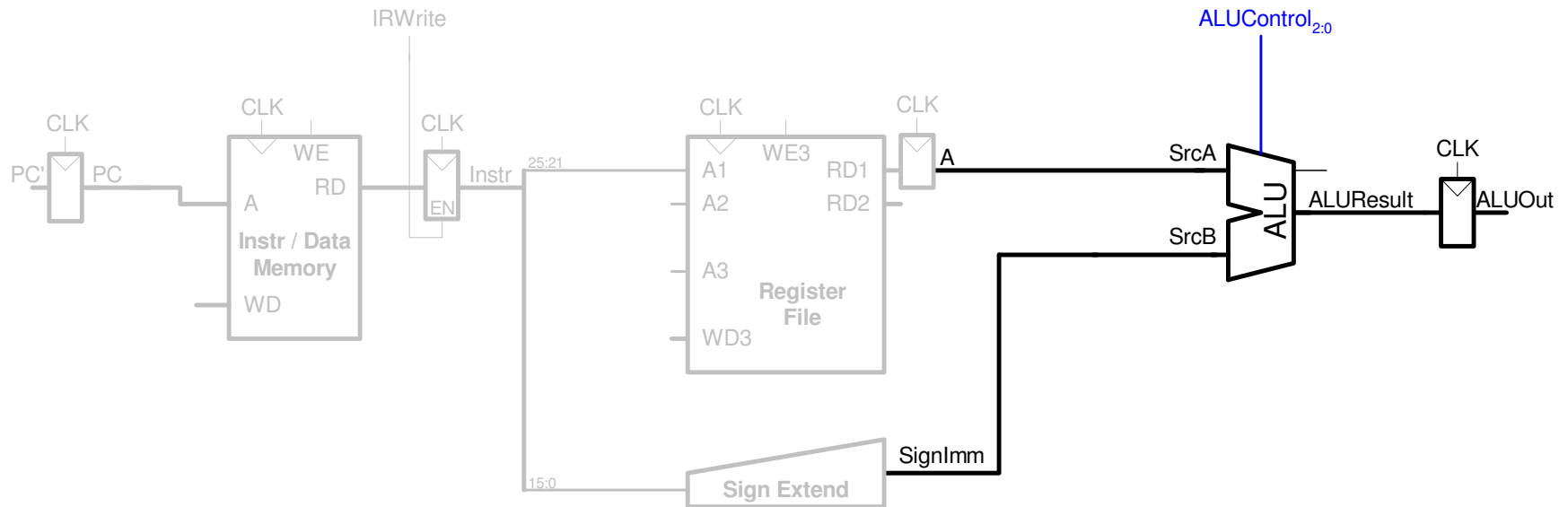
# Processador MIPS Multicycle

- Sign-Extend o Imediato



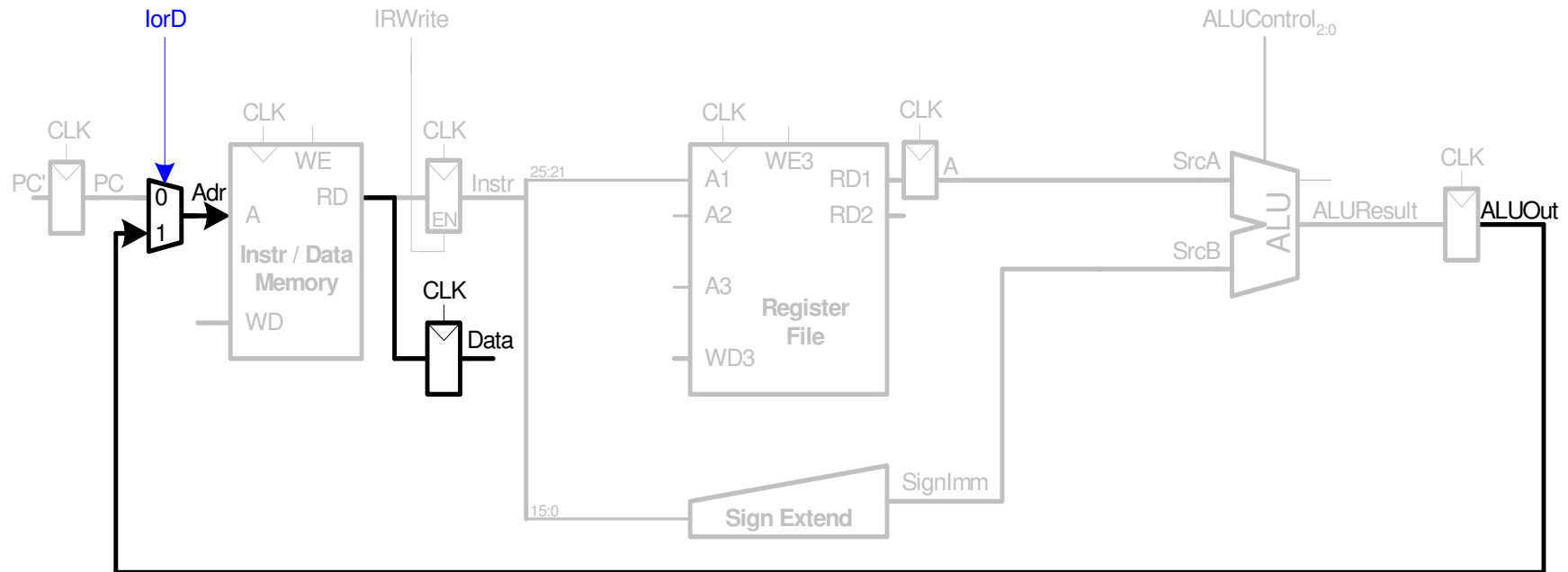
# Processador MIPS Multicycle

- Soma do Base Address ao Offset



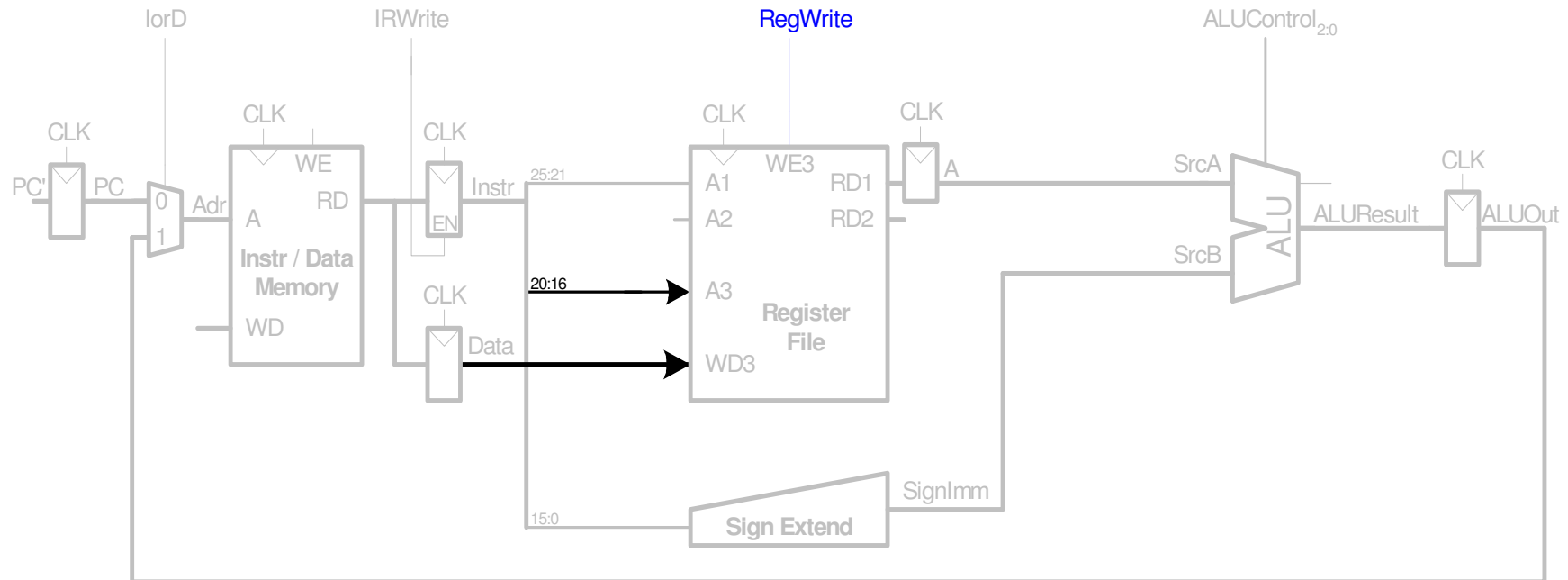
# Processador MIPS Multicycle

- Carga do dado a partir da Memória



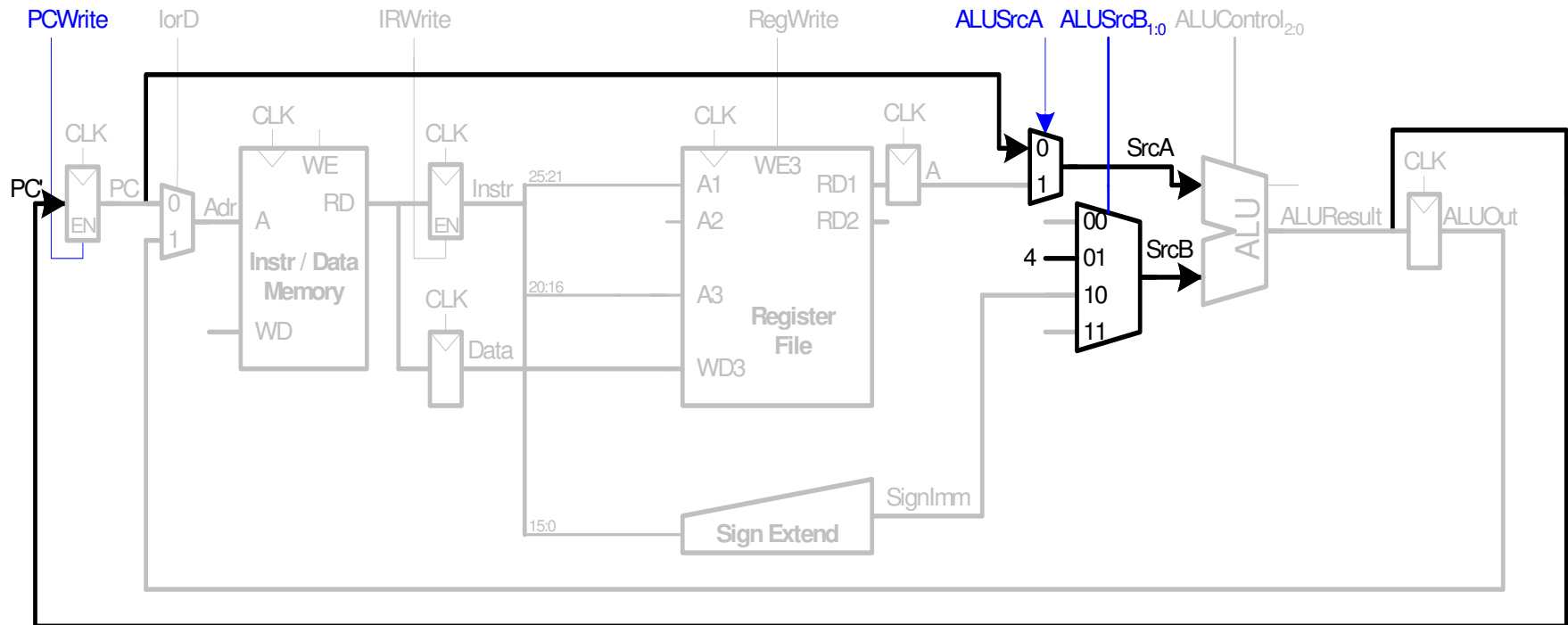
# Processador MIPS Multicycle

- Escrita do Dado no Register File



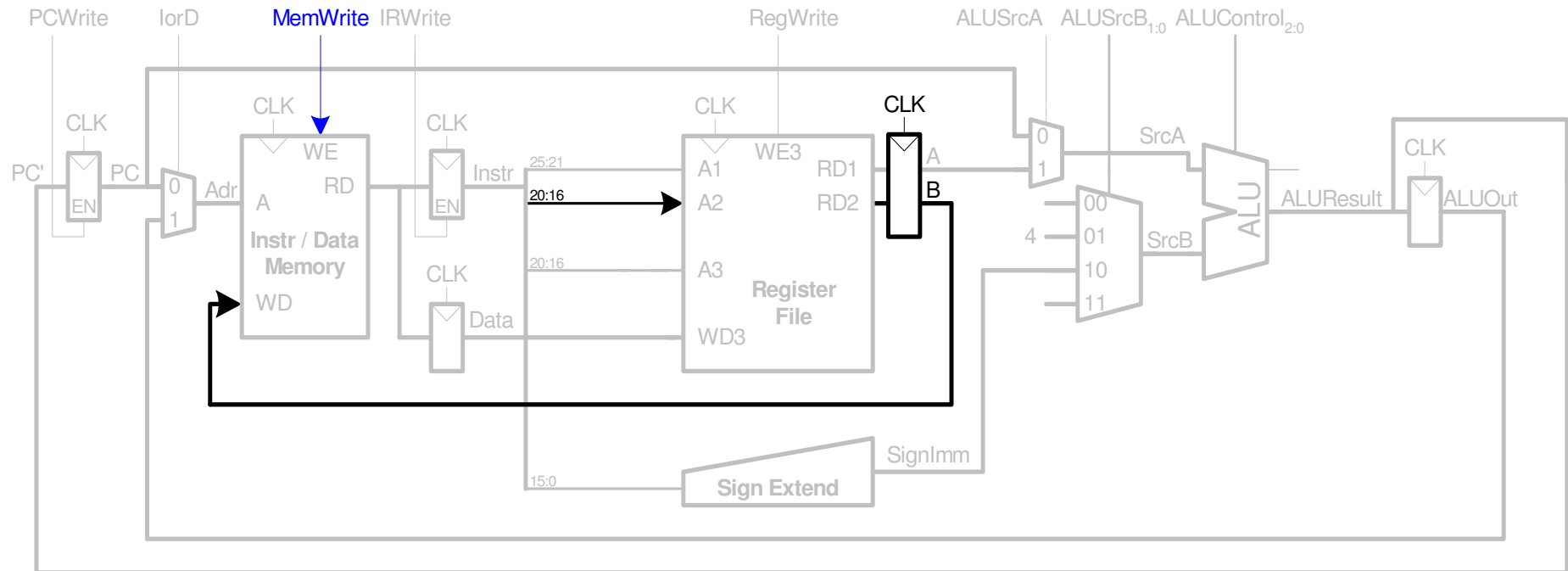
# Processador MIPS Multicycle

- Incremento do PC de 4



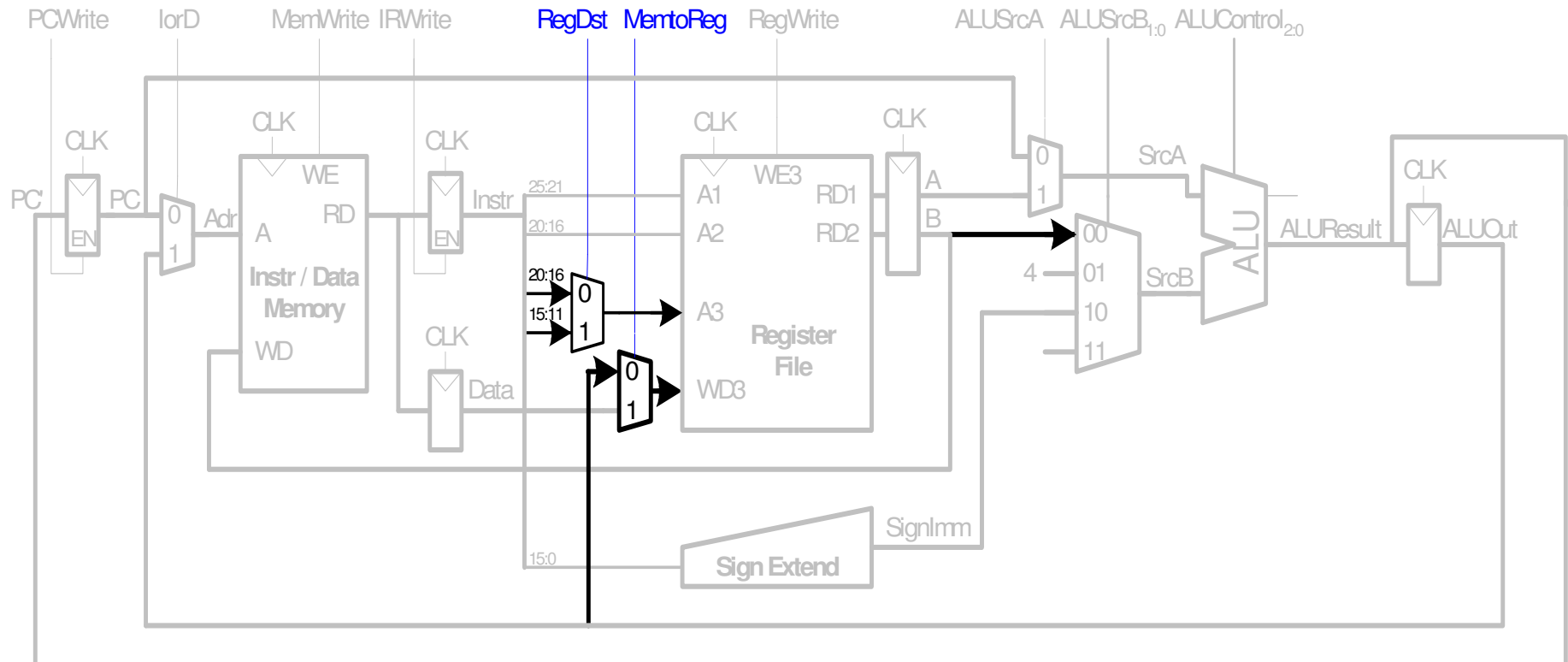
# Processador MIPS Multicycle

- Datapath também para sw



# Processador MIPS Multicycle

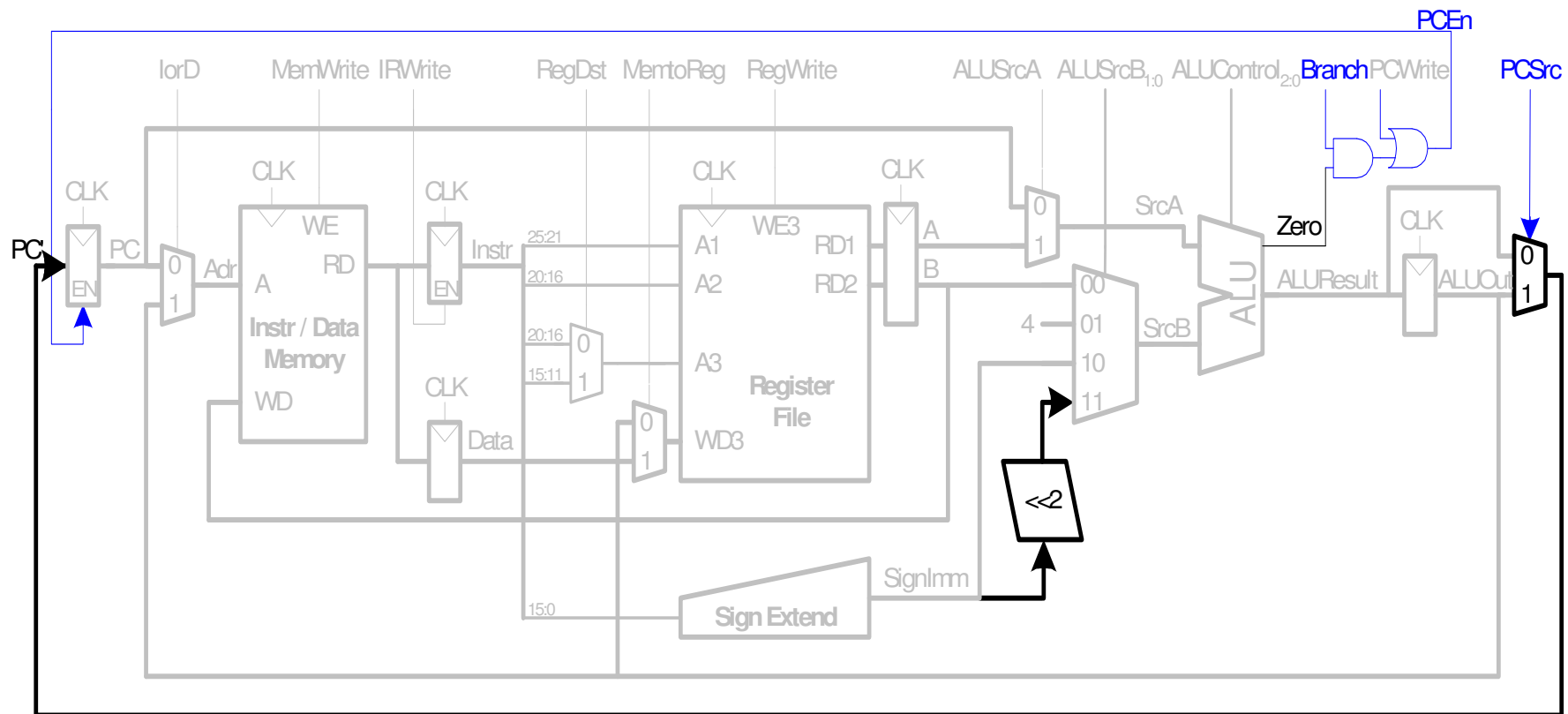
- Datapath para instruções R-Type





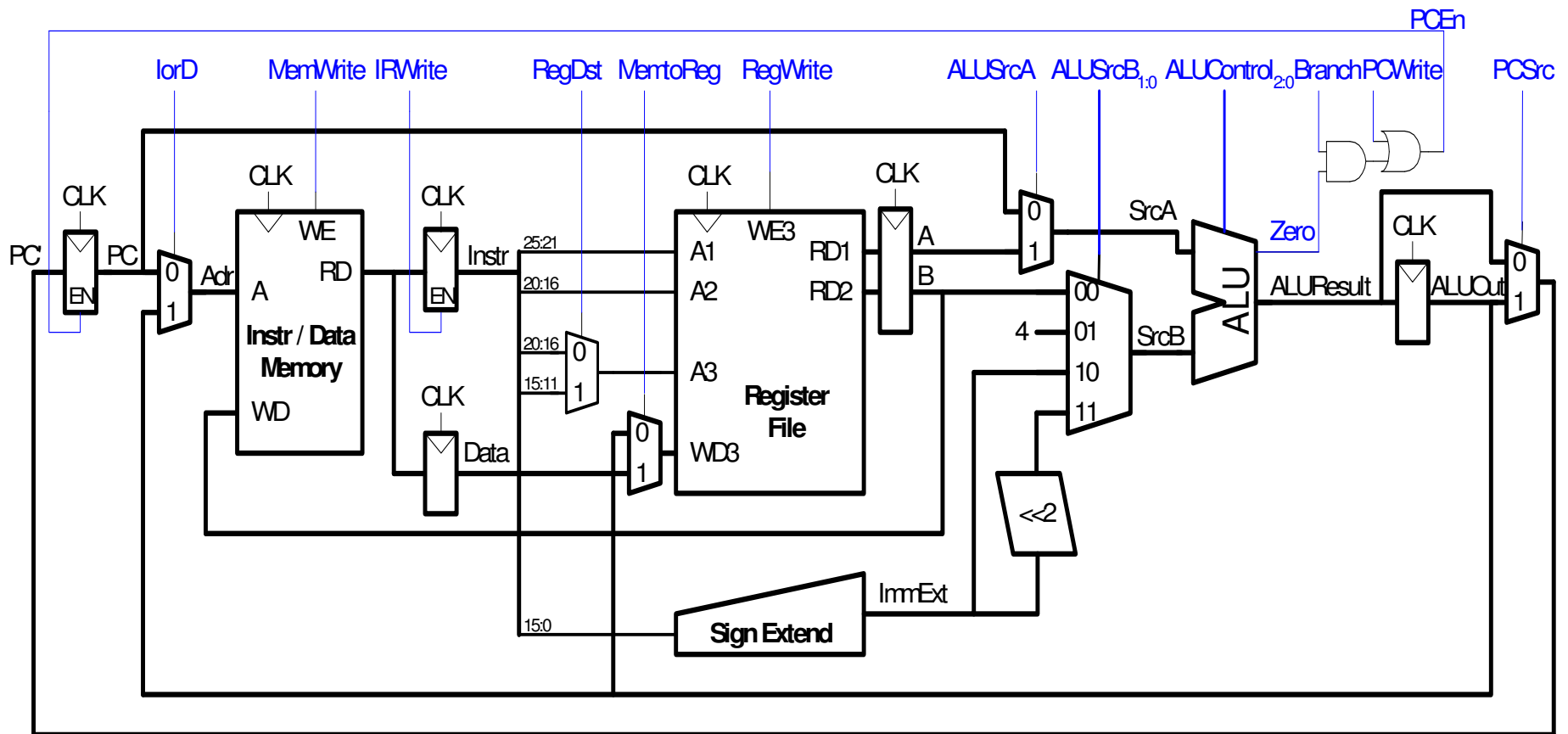
# Processador MIPS Multicycle

- Datapath para beq



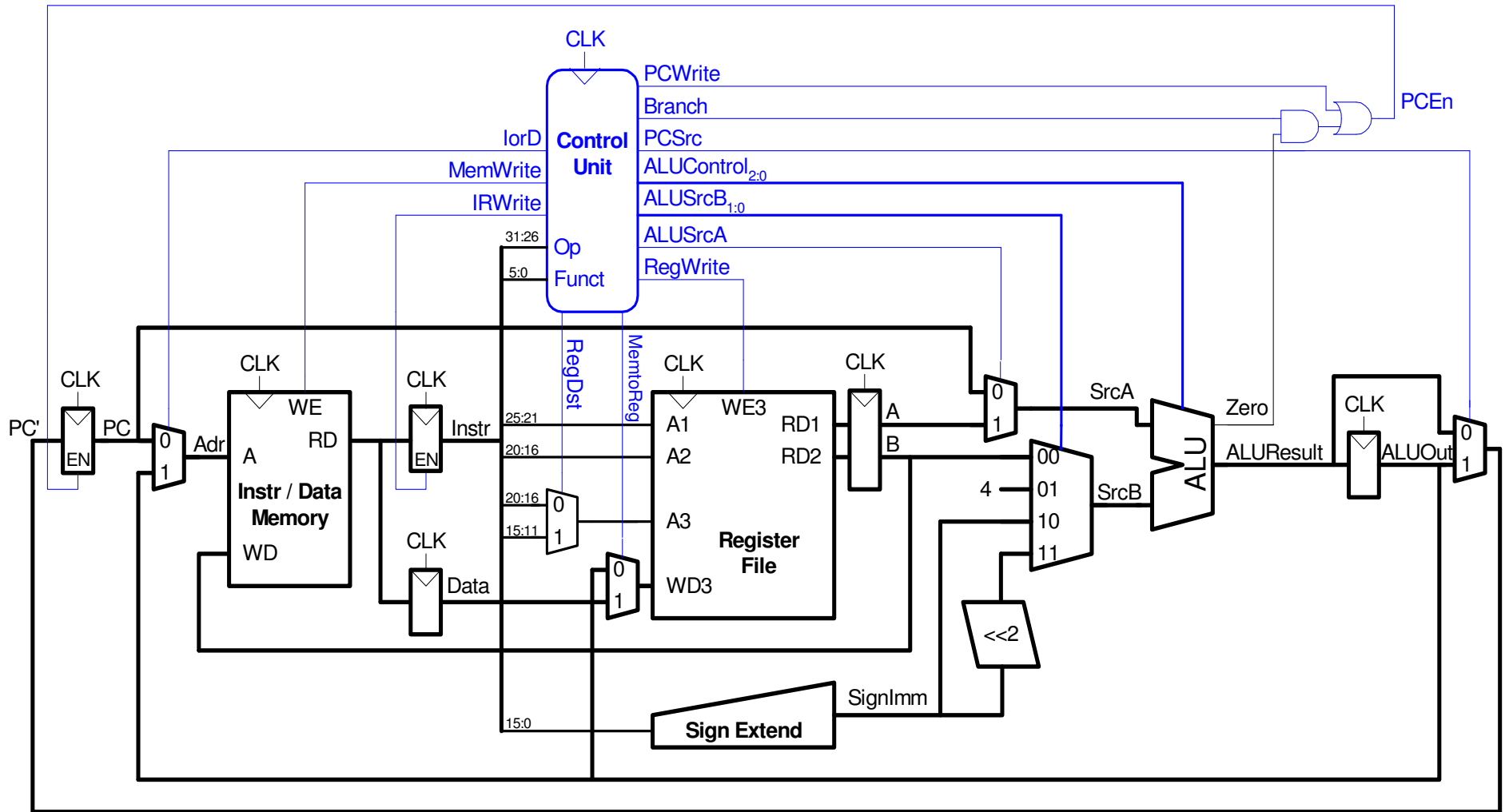
# Processador MIPS Multicycle

- Datapath para o Processador Multicycle



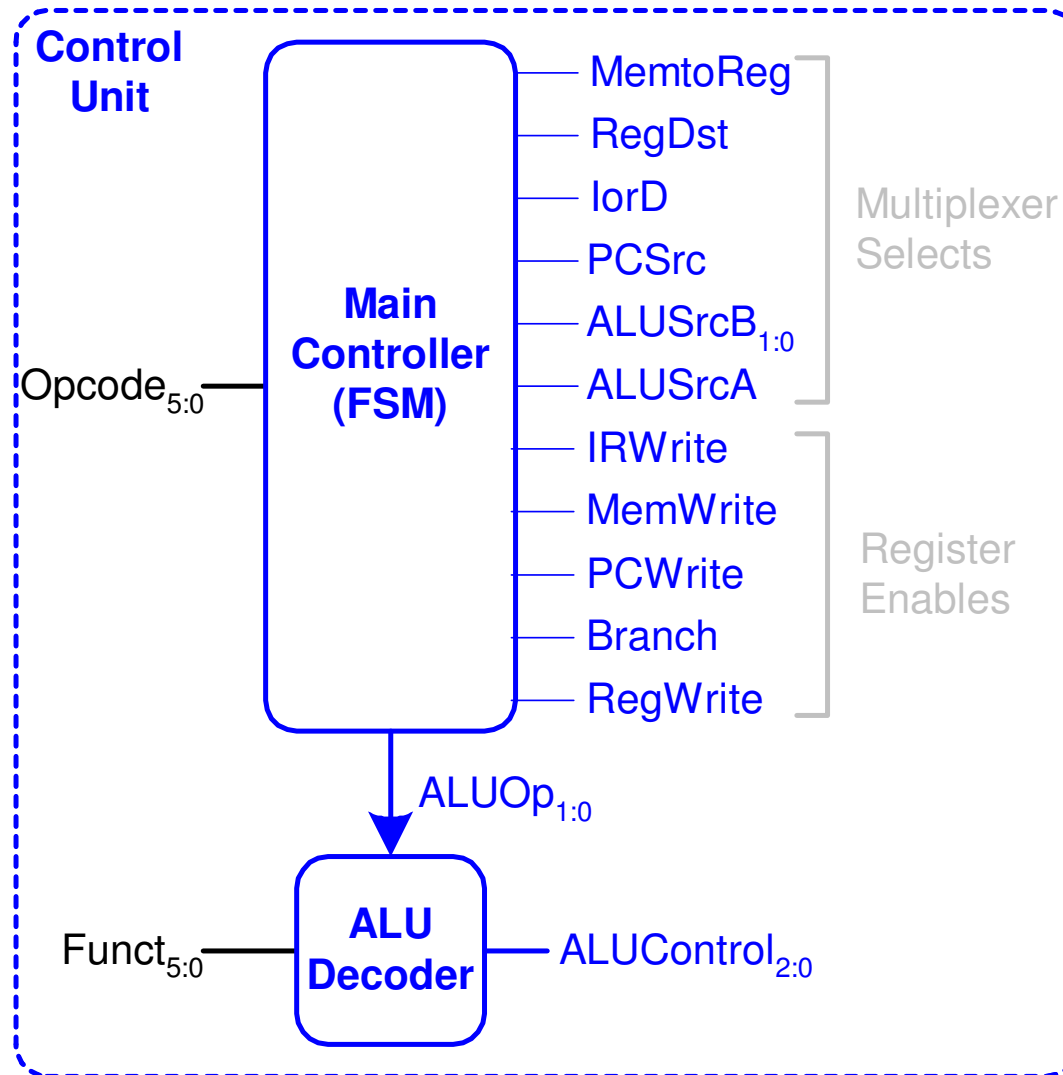
# Processador MIPS Multicycle

- Controle para o Processador Multicycle



# Processador MIPS Multicycle

- Unidade de controle



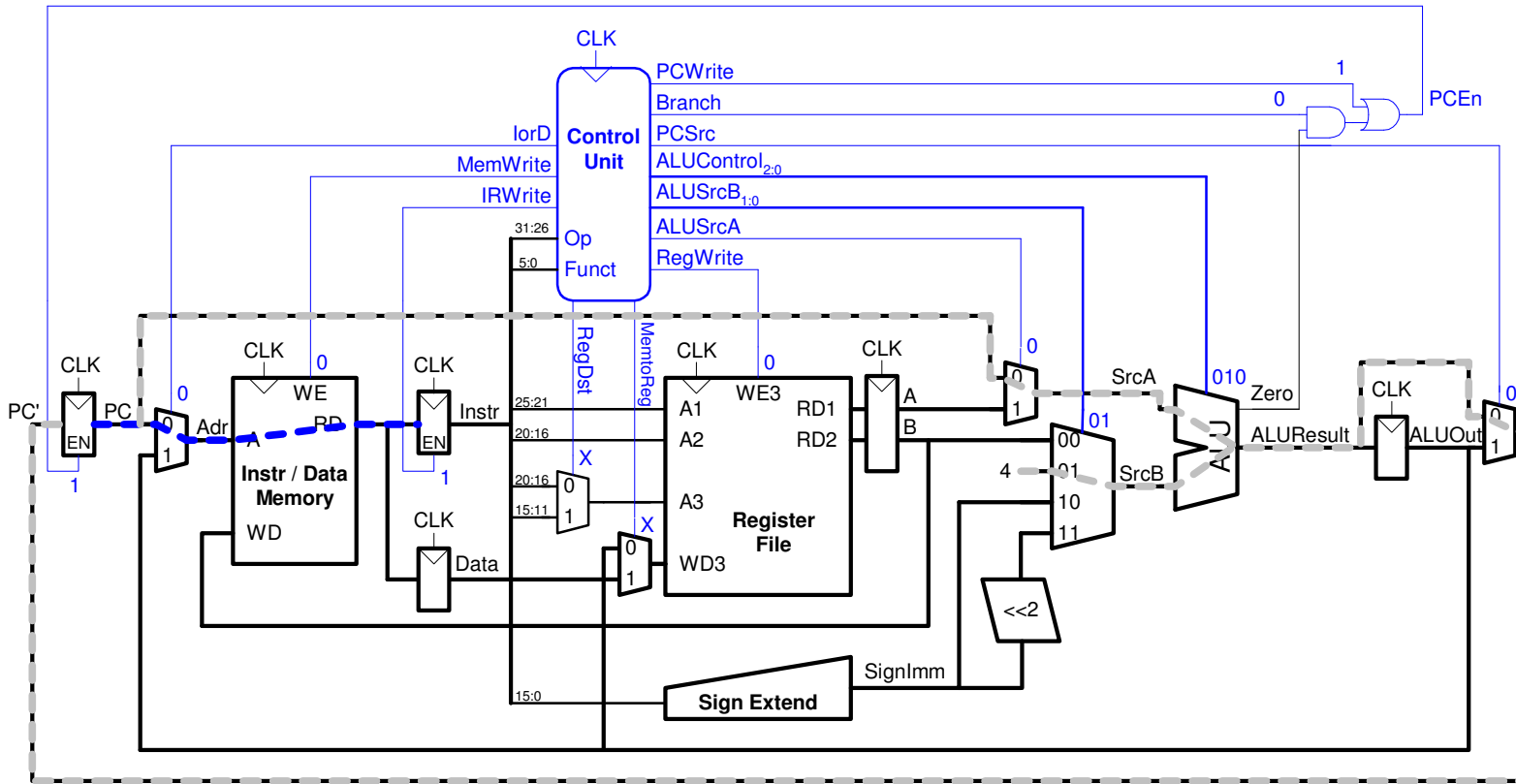
# Processador MIPS Multicycle

- Unidade de Controle
  - Finite state machine
  - Diagrama de transição de estados para lw

# Processador MIPS Multicycle

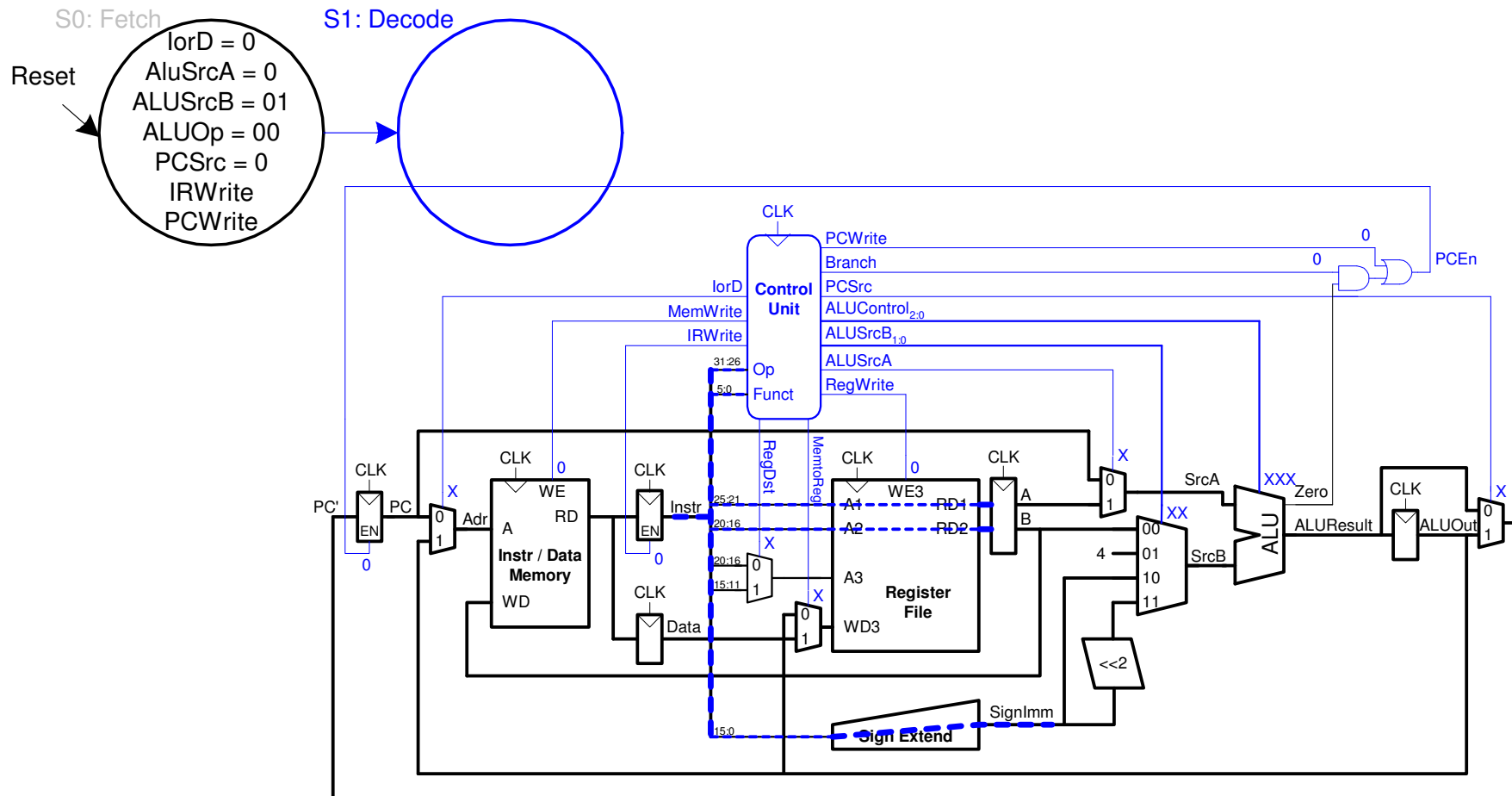
- Fetch da Instrução

S0: Fetch  
 Reset  
 lorD = 0  
 AluSrcA = 0  
 ALUSrcB = 01  
 ALUOp = 00  
 PCSrc = 0  
 IRWrite  
 PCWrite



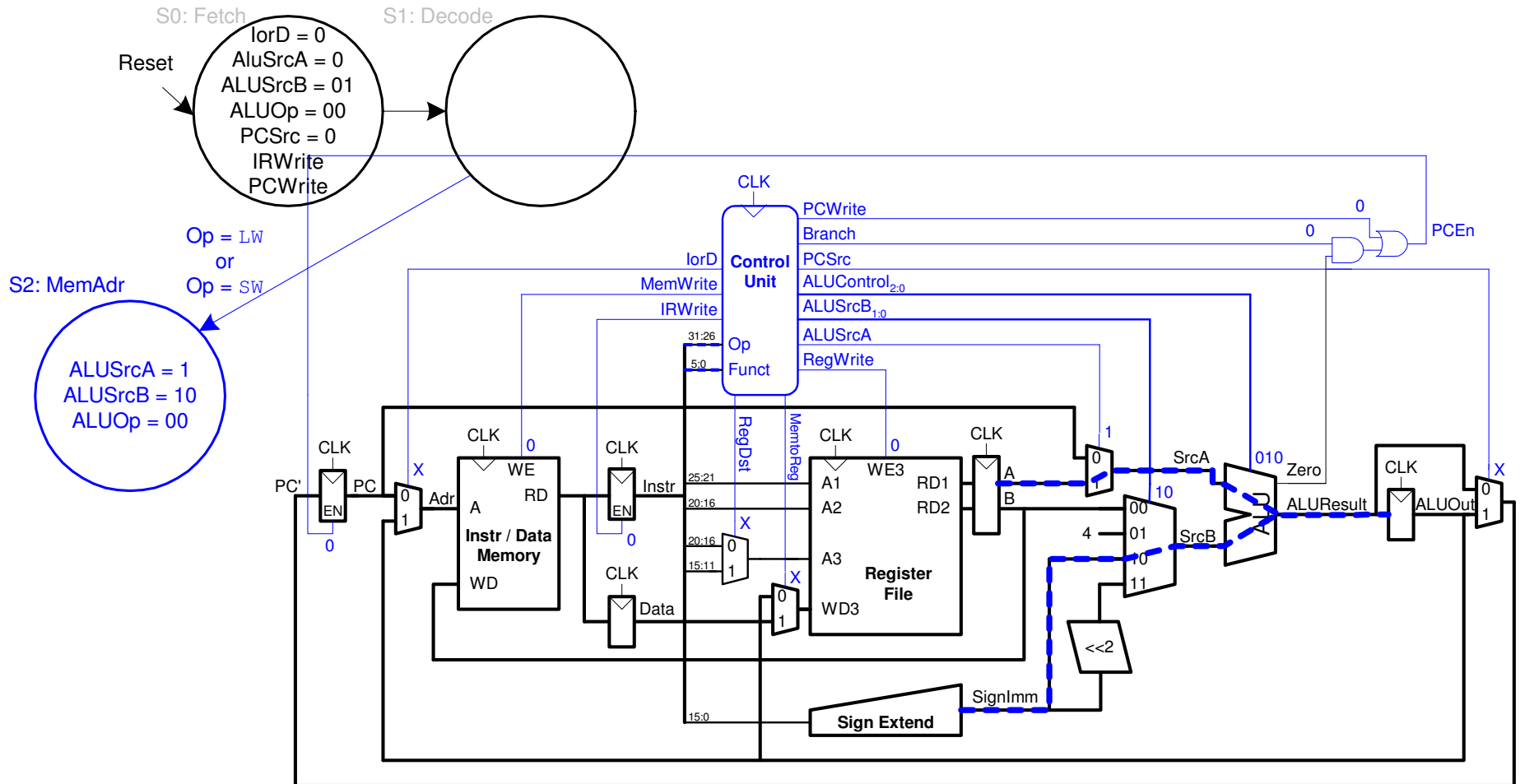
# Processador MIPS Multicycle

- Decodificação da Instrução



# Processador MIPS Multicycle

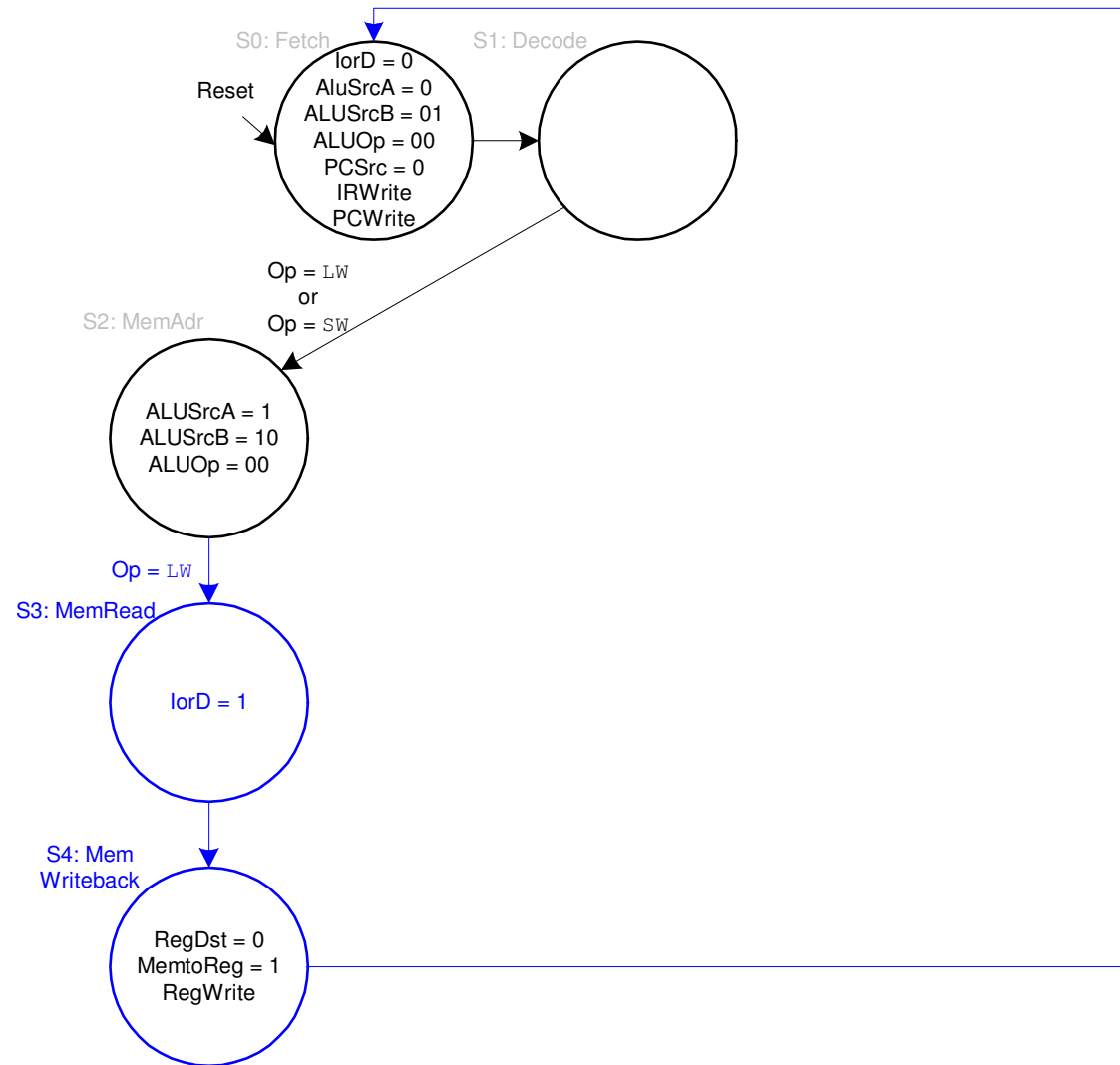
- Cálculo do endereço efetivo





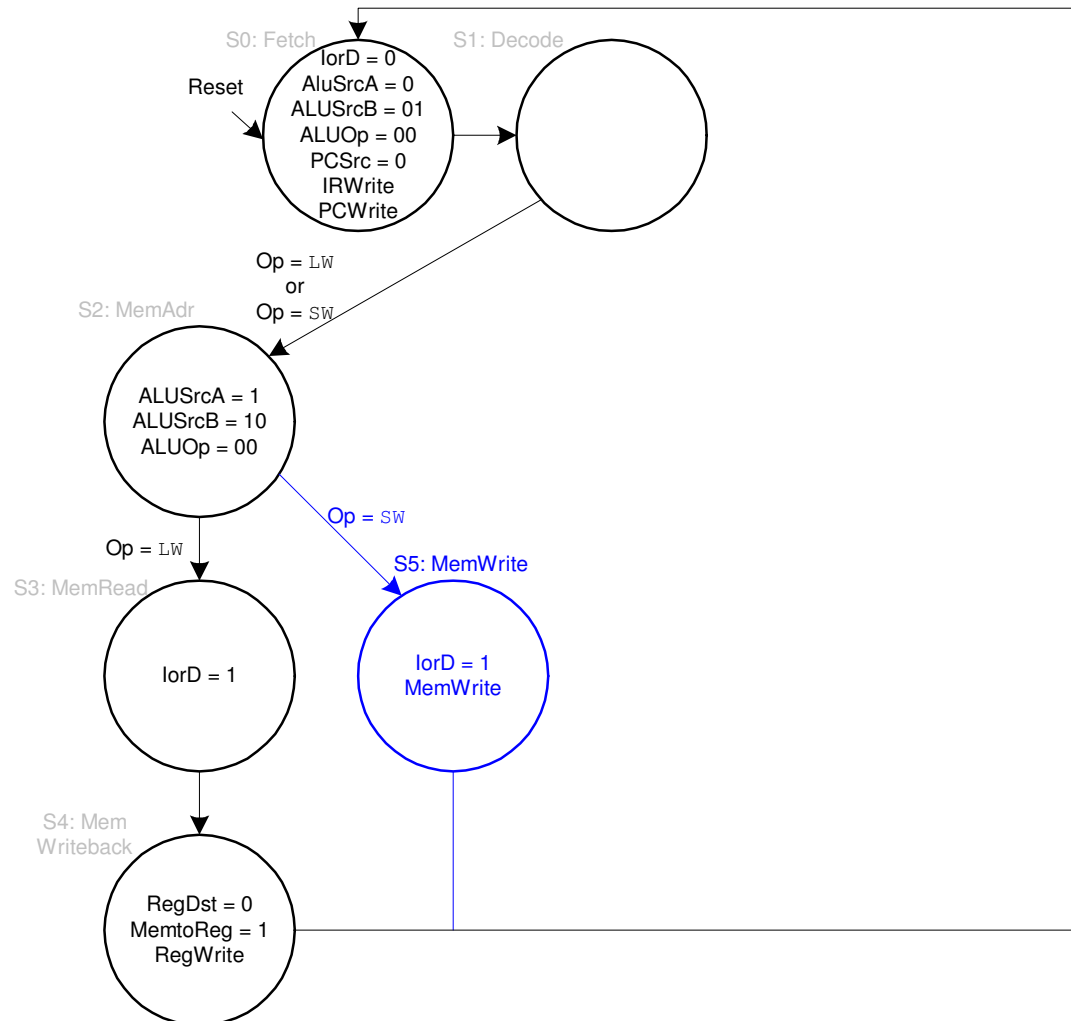
# Processador MIPS Multicycle

- Leitura da Memória e Escrita no RF



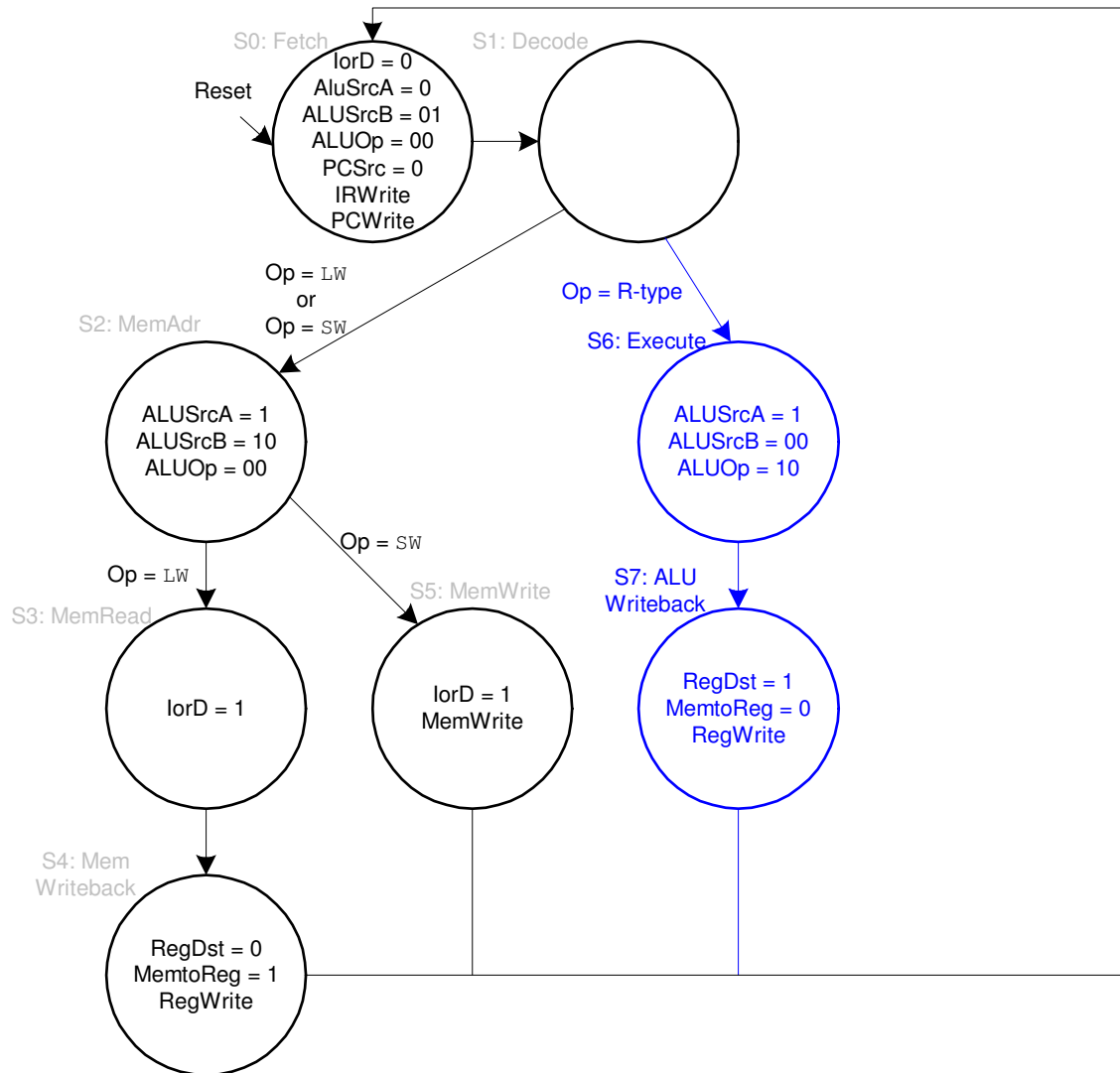
# Processador MIPS Multicycle

- Diagrama de transição de estados para sw



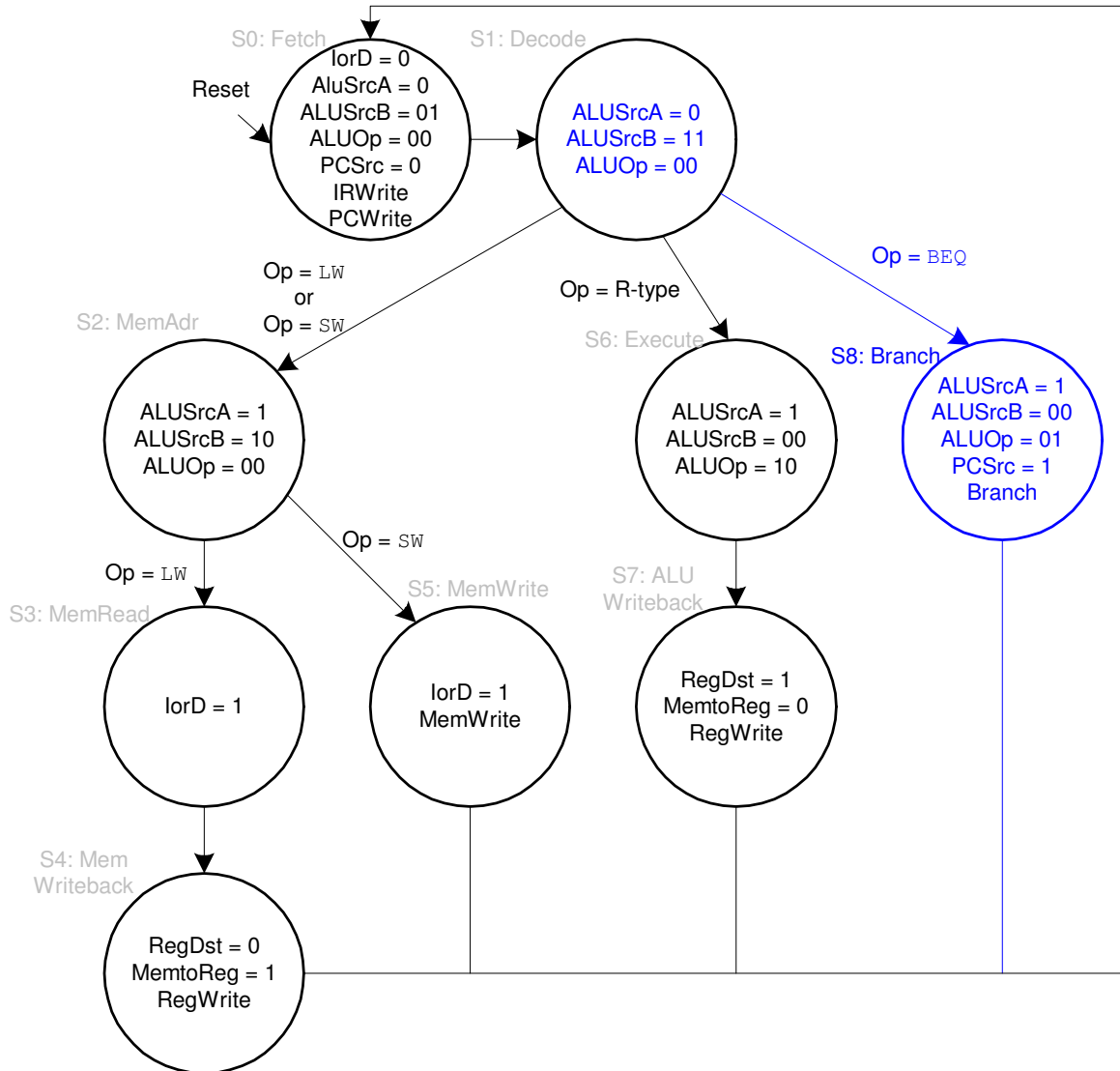
# Processador MIPS Multicycle

- Diagrama de transição de estados para R-type



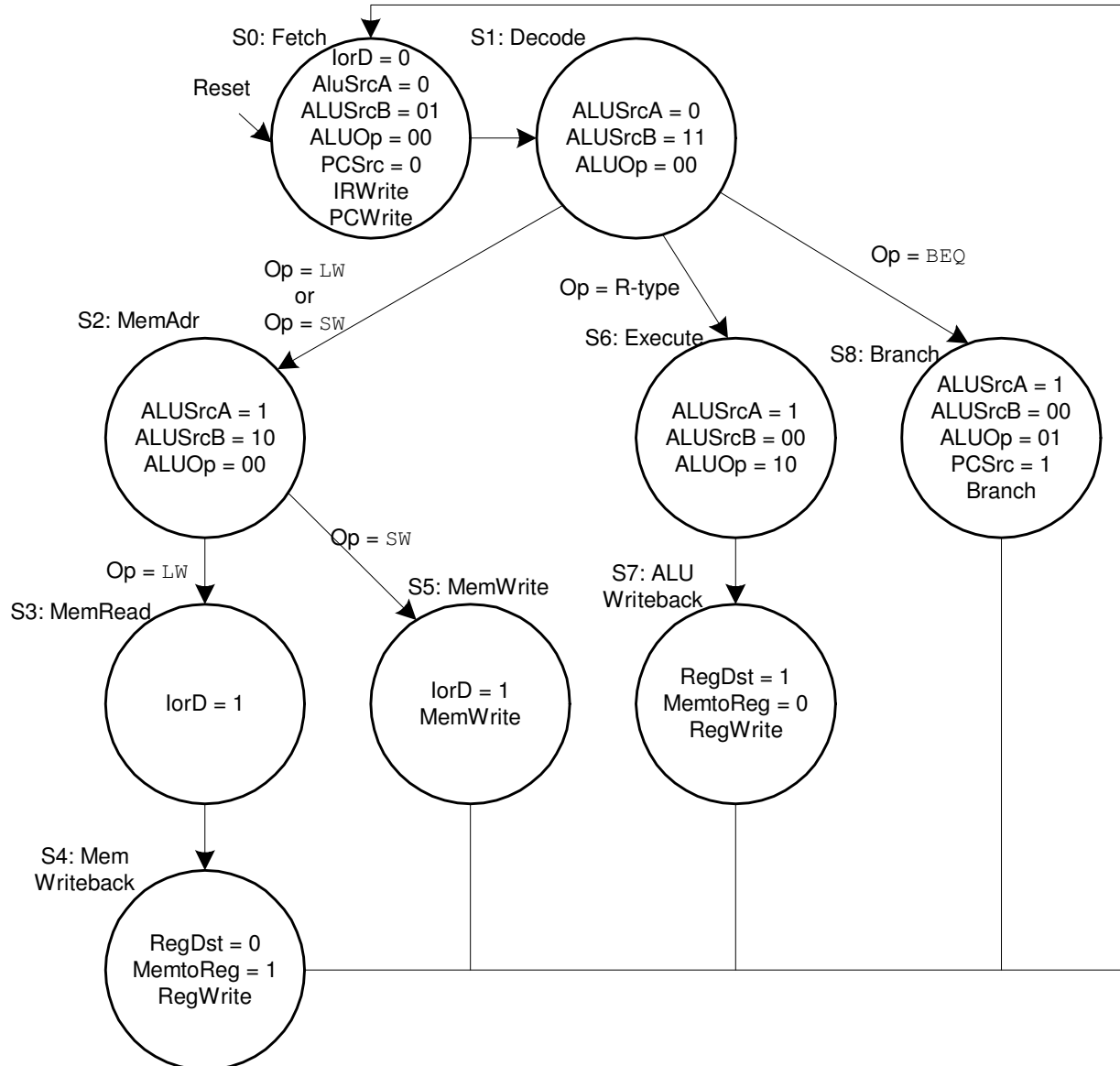
# Processador MIPS Multicycle

- Diagrama de transição de estados para beq



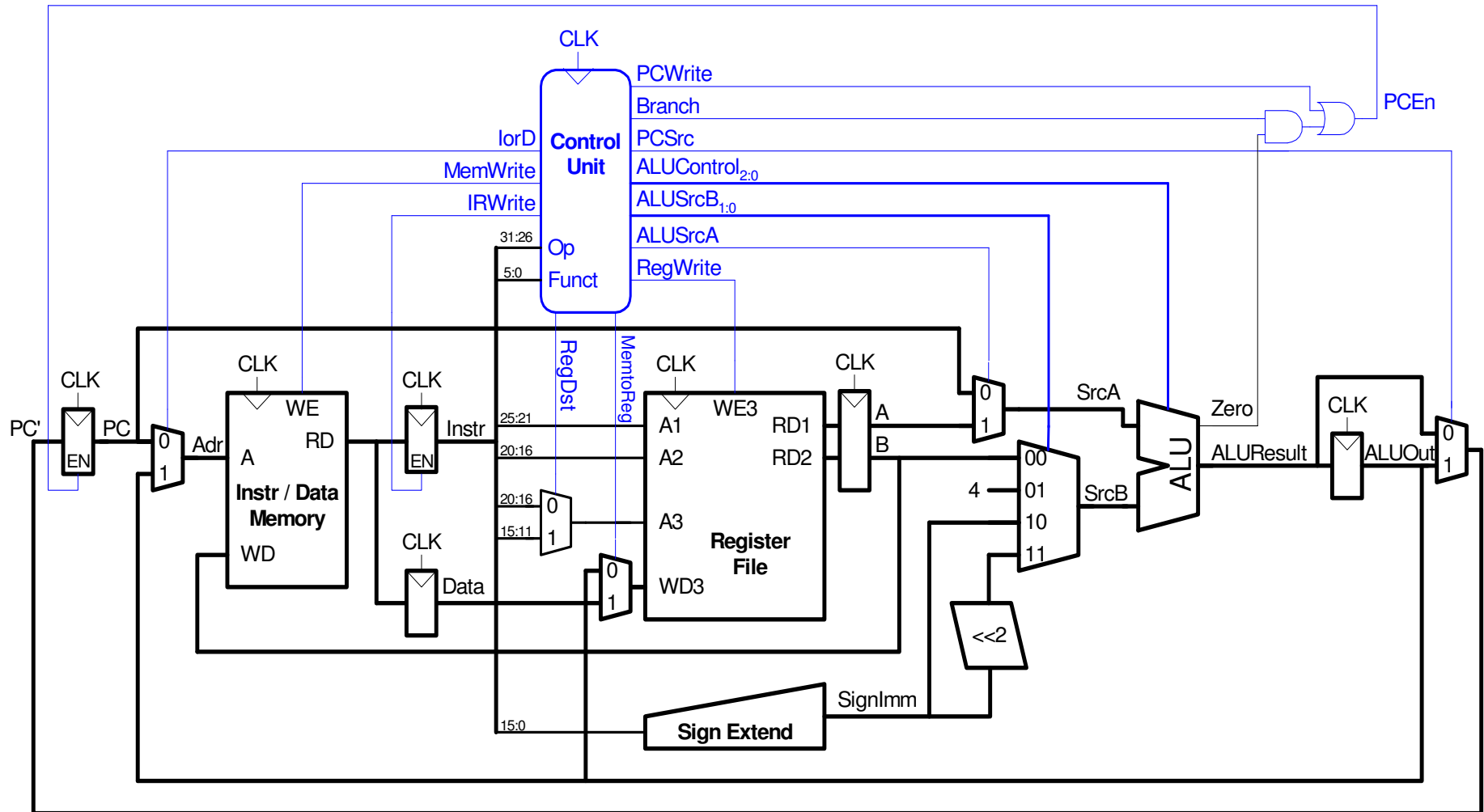
# Processador MIPS Multicycle

- FSM de Controle



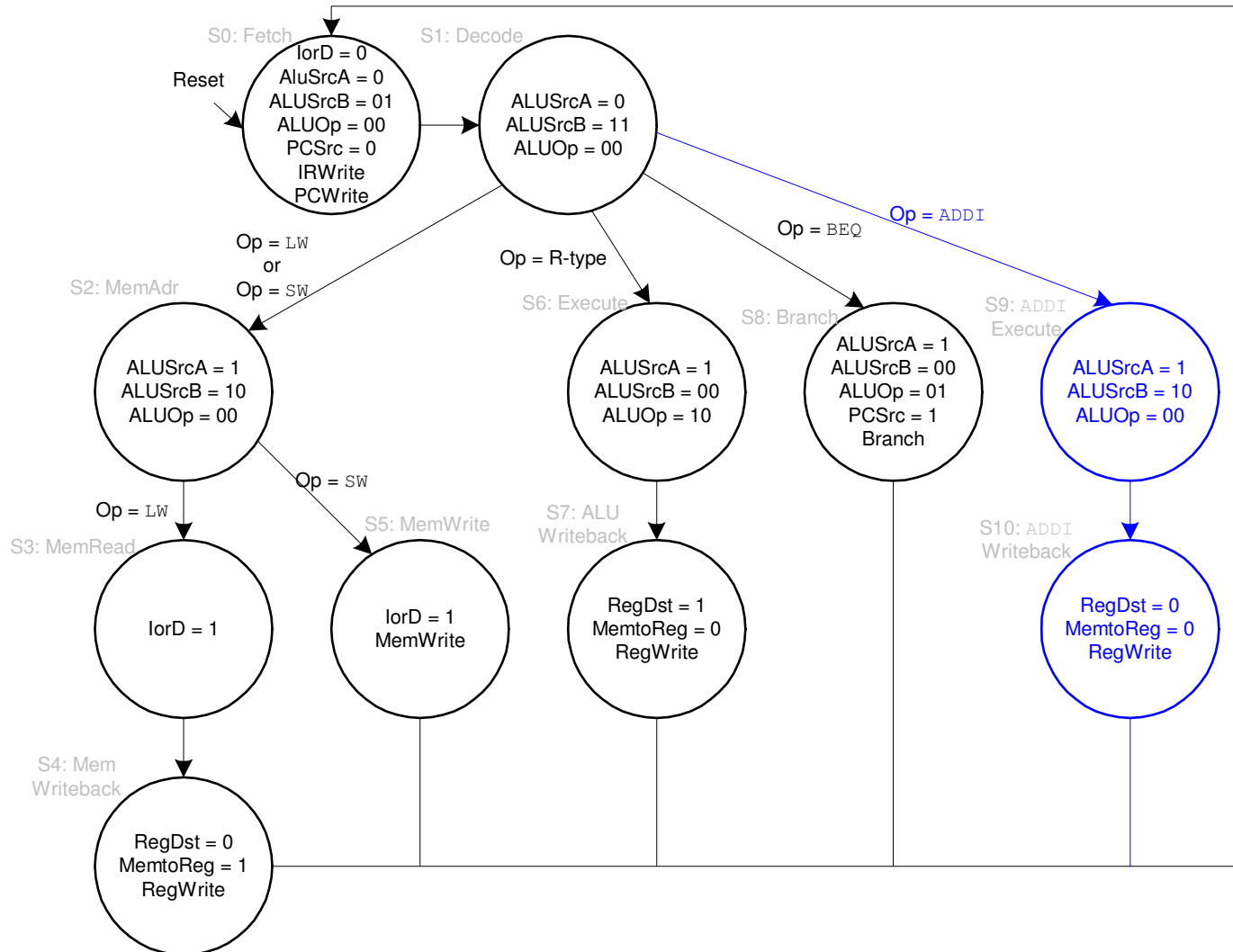
# Processador MIPS Multicycle

- Extendendo a Funcionalidade para addi



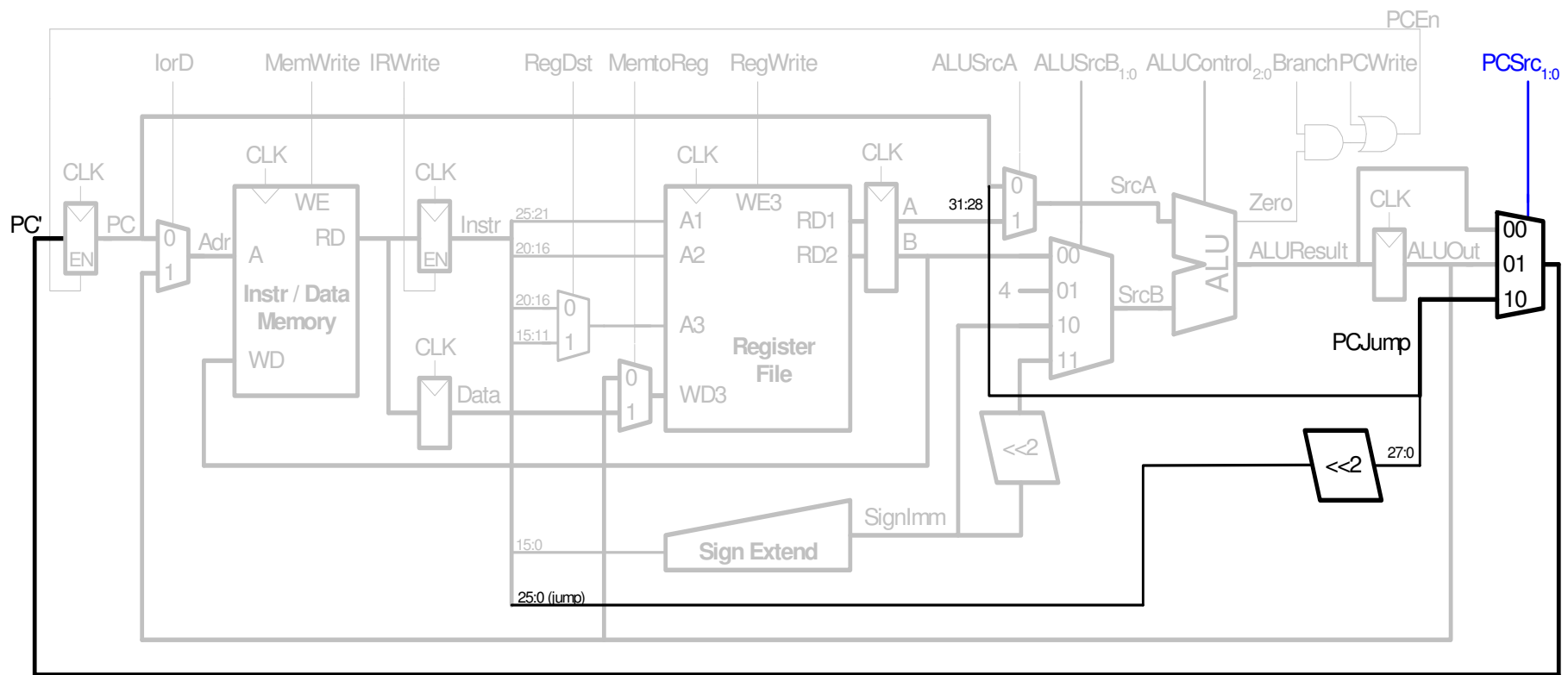
# Processador MIPS Multicycle

- FSM de Controle para addi



# Processador MIPS Multicycle

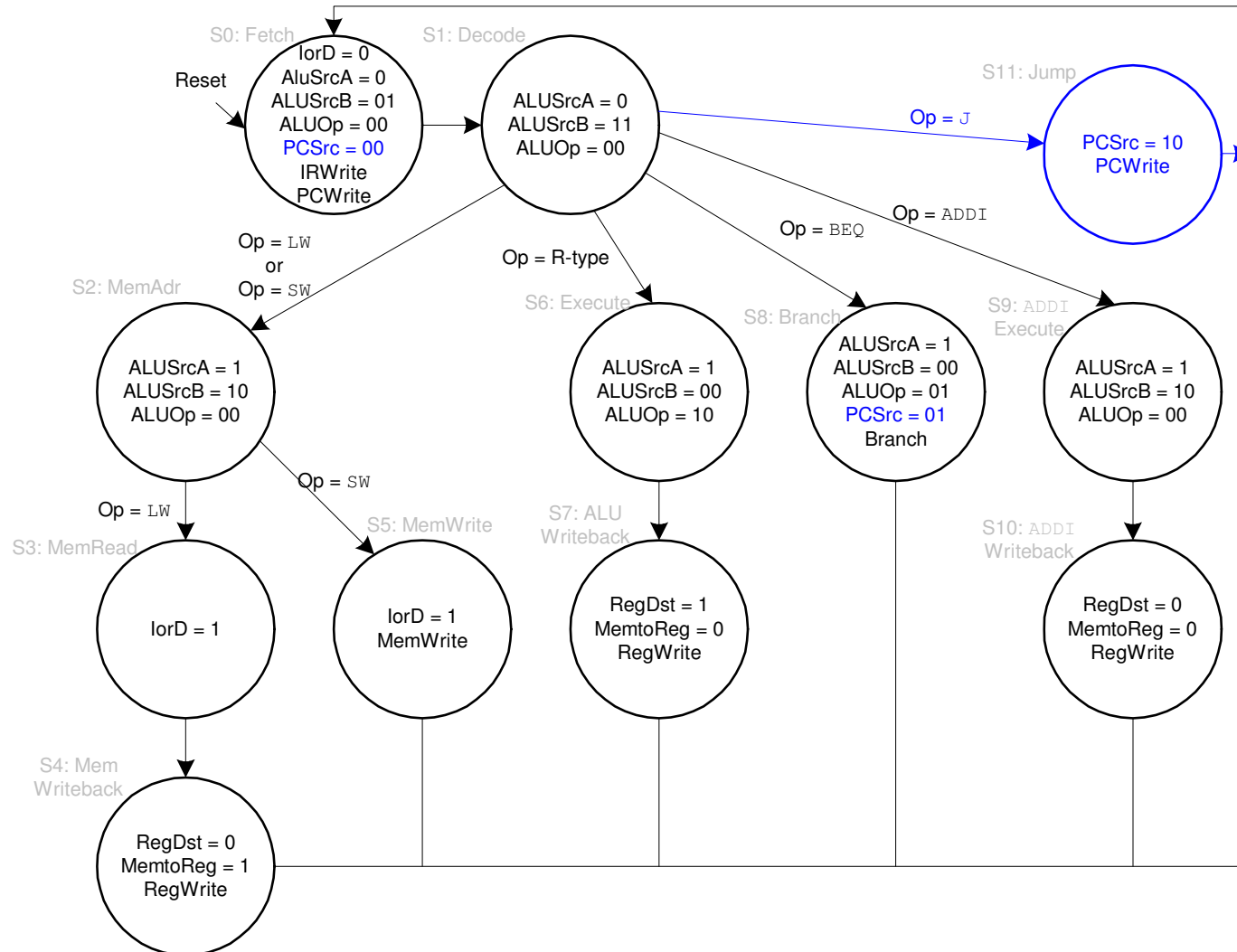
- Extendendo a Funcionalidade para j





# Processador MIPS Multicycle

- FSM de Controle para j



# Processador MIPS Multicycle

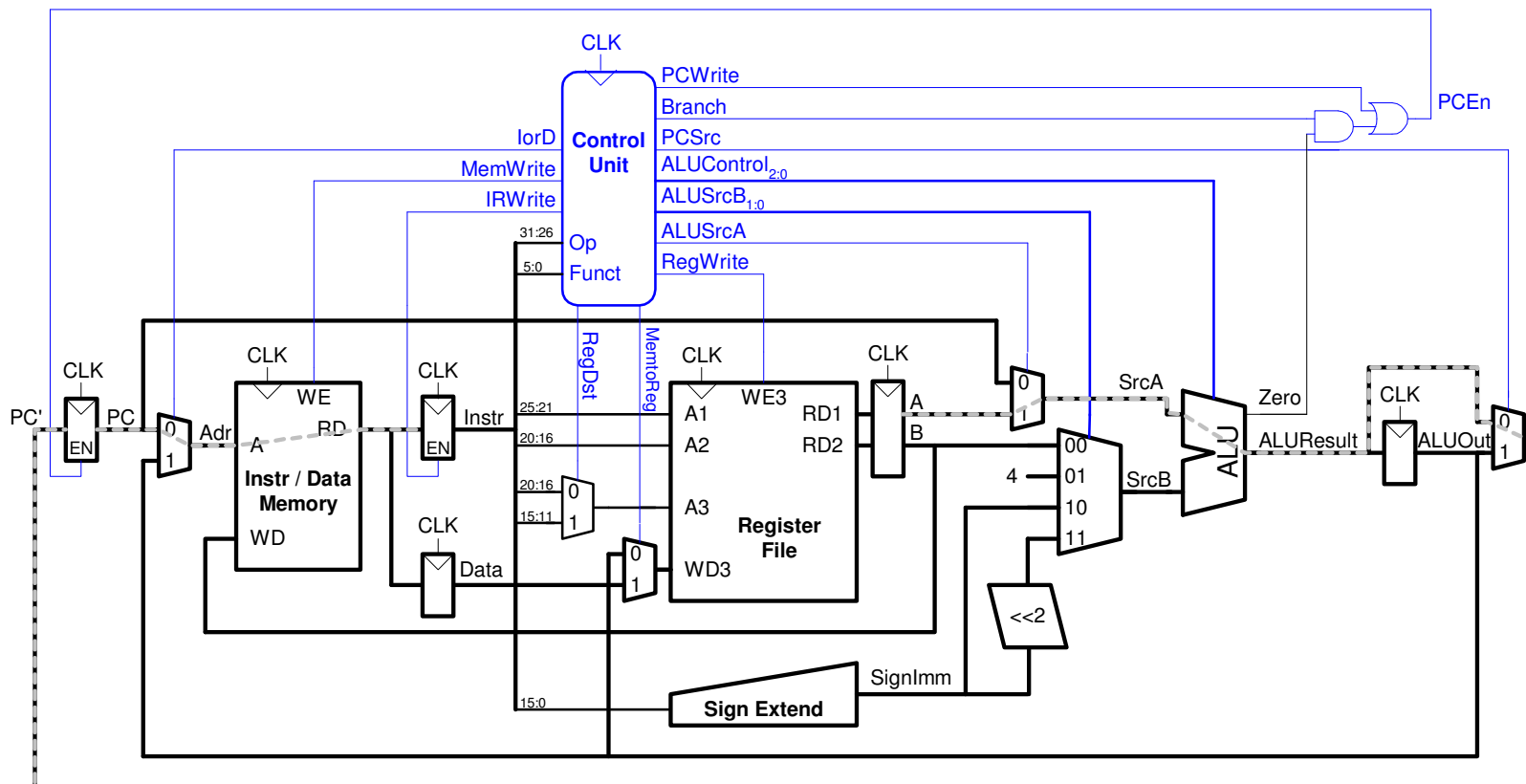
- **Desempenho: Quão rápido é o processador?**
- **Instruções gastam números diferentes de ciclos:**
  - 3 ciclos: beq, j
  - 4 ciclos: R-Type, sw, addi
  - 5 ciclos: lw
- **O CPI deve ser a média ponderada**
- **SPECINT2000 benchmark:**
  - 25% loads
  - 10% stores
  - 11% branches
  - 2% jumps
  - 52% R-type

$$\text{Average CPI} = (0.11 + 0.2)(3) + (0.52 + 0.10)(4) + (0.25)(5) = 4.12$$

# Processador MIPS Multicycle

- Caminho Crítico

$$T_c = t_{pcq\_PC} + t_{mux} + \max(t_{ALU} + t_{mux}, t_{mem}) + t_{setup}$$



# Processador MIPS Multicycle

Element	Parameter	Delay (ps)
Register clock-to-Q	$t_{pcq\_PC}$	30
Register setup	$t_{setup}$	20
Multiplexer	$t_{mux}$	25
ALU	$t_{ALU}$	200
Memory read	$t_{mem}$	250
Register file read	$t_{RFread}$	150
Register file setup	$t_{RFsetup}$	20

$$\begin{aligned}T_c &= t_{pcq\_PC} + t_{mux} + \max(t_{ALU} + t_{mux}, t_{mem}) + t_{setup} \\ &= t_{pcq\_PC} + t_{mux} + t_{mem} + t_{setup} \\ &= [30 + 25 + 250 + 20] \text{ ps} \\ &= 325 \text{ ps}\end{aligned}$$

# Processador MIPS Multicycle

- Para um programa que executa 100 bilhões de instruções em um MIPS multicycle,
- $CPI = 4.12$
- $T_c = 325 \text{ ps}$

$$\begin{aligned} \text{Execution Time} &= (\# \text{ instructions}) \times CPI \times T_c \\ &= (100 \times 10^9)(4.12)(325 \times 10^{-12}) \\ &= 133.9 \text{ seconds} \end{aligned}$$

- Ele é mais **lento** do que o MIPS single-cycle (95 segundos). Por que?
  - Os passos não tem o mesmo tamanho
  - Overhead do seqüenciamento de cada passo ( $t_{pcq} + t_{\text{setup}} = 50 \text{ ps}$ )



# Revisão: Processador MIPS Multicycle

