

# Arquitetura de um processador

## 1. Caminho de dados de um processador

Prof. John L. Gardenghi  
*Adaptado dos slides do livro*

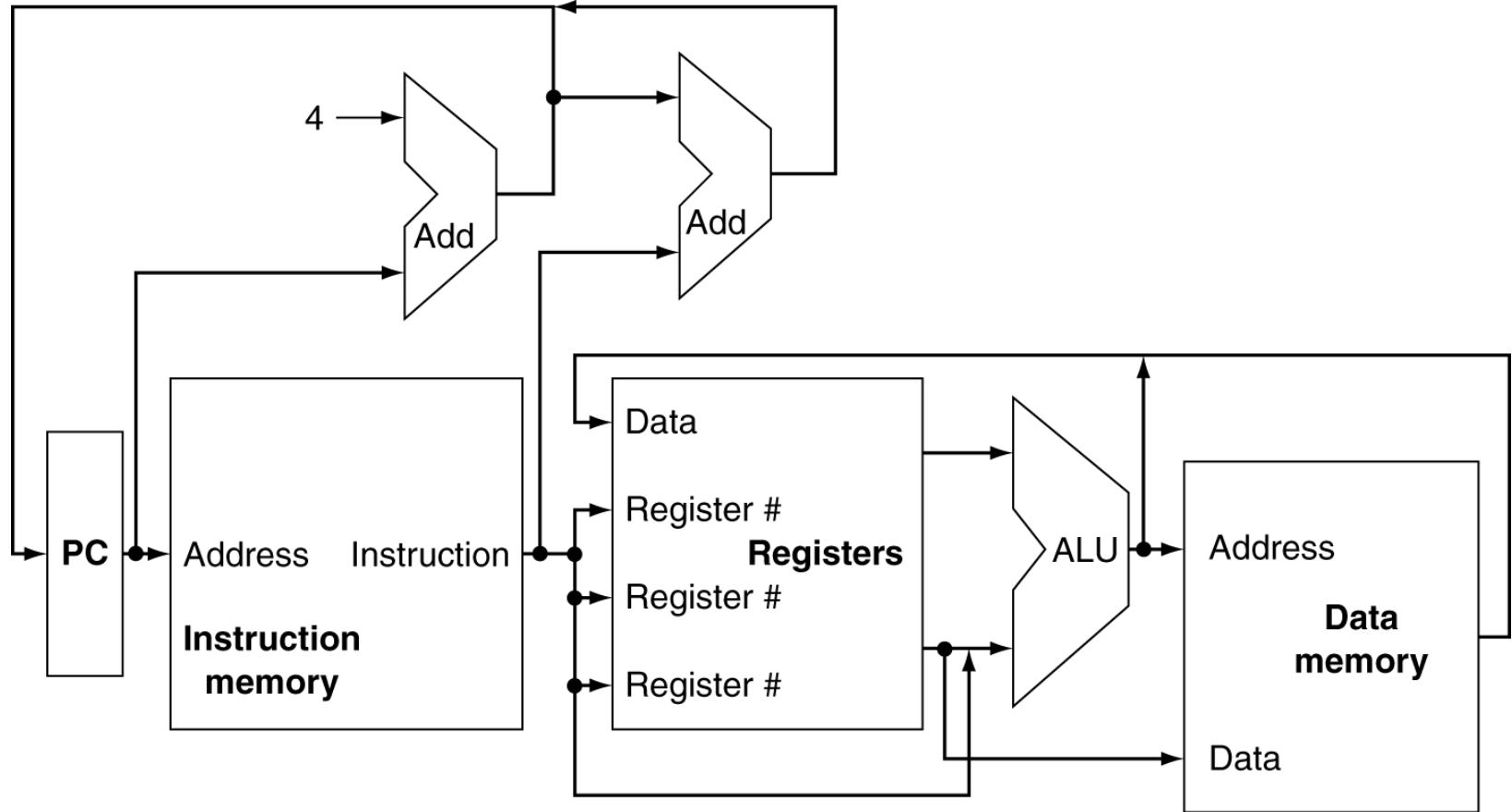
# Introdução

- Nesta parte, estudaremos a implementação de um processador MIPS
  - Versão simplificada (monociclo)
  - Versão com pipeline
- Analisaremos o seguinte subconjunto de instruções
  - Acesso à memória: lw, sw
  - Lógicas e aritméticas: add, sub, and, or, slt
  - Desvio condicional: beq, j

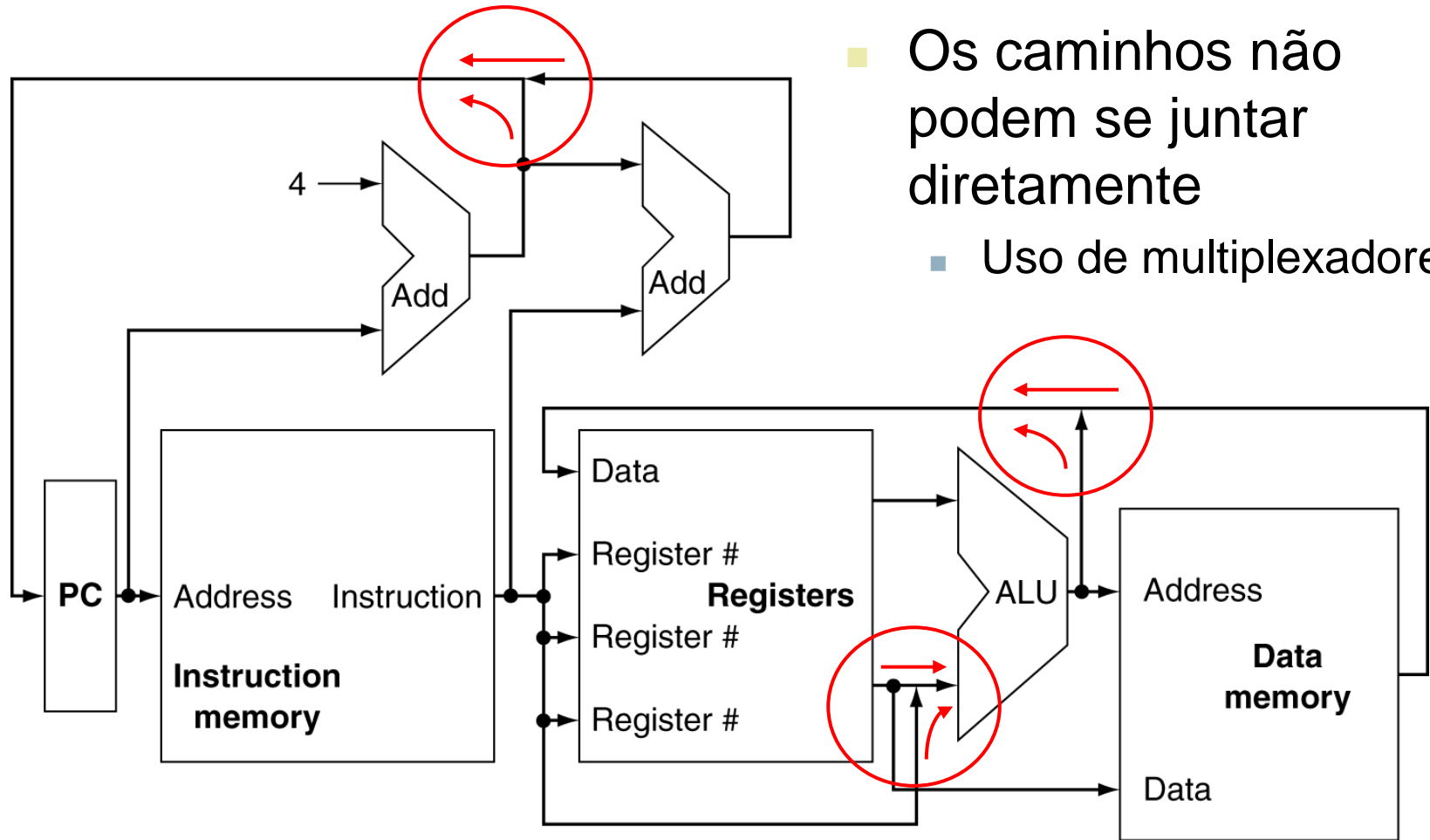
# Execução de uma instrução

- Ciclo fetch → decode → execute.
  - PC → acesso à memória de instruções, obtém a instrução (fetch)
  - Decodifica a instrução e obtém os dados necessários para a execução
    - Acesso aos registradores
  - Dependendo da classe da instrução:
    - Usa uma ULA para cálculos
      - Instruções aritméticas: resultado da operação
      - Instruções de acesso à memória: cálculo do endereço
      - Instruções de desvio: endereço do desvio
    - Faz acesso à memória
    - $PC \leftarrow$  destino do desvio ou  $PC + 4$

# Visão geral do CPU

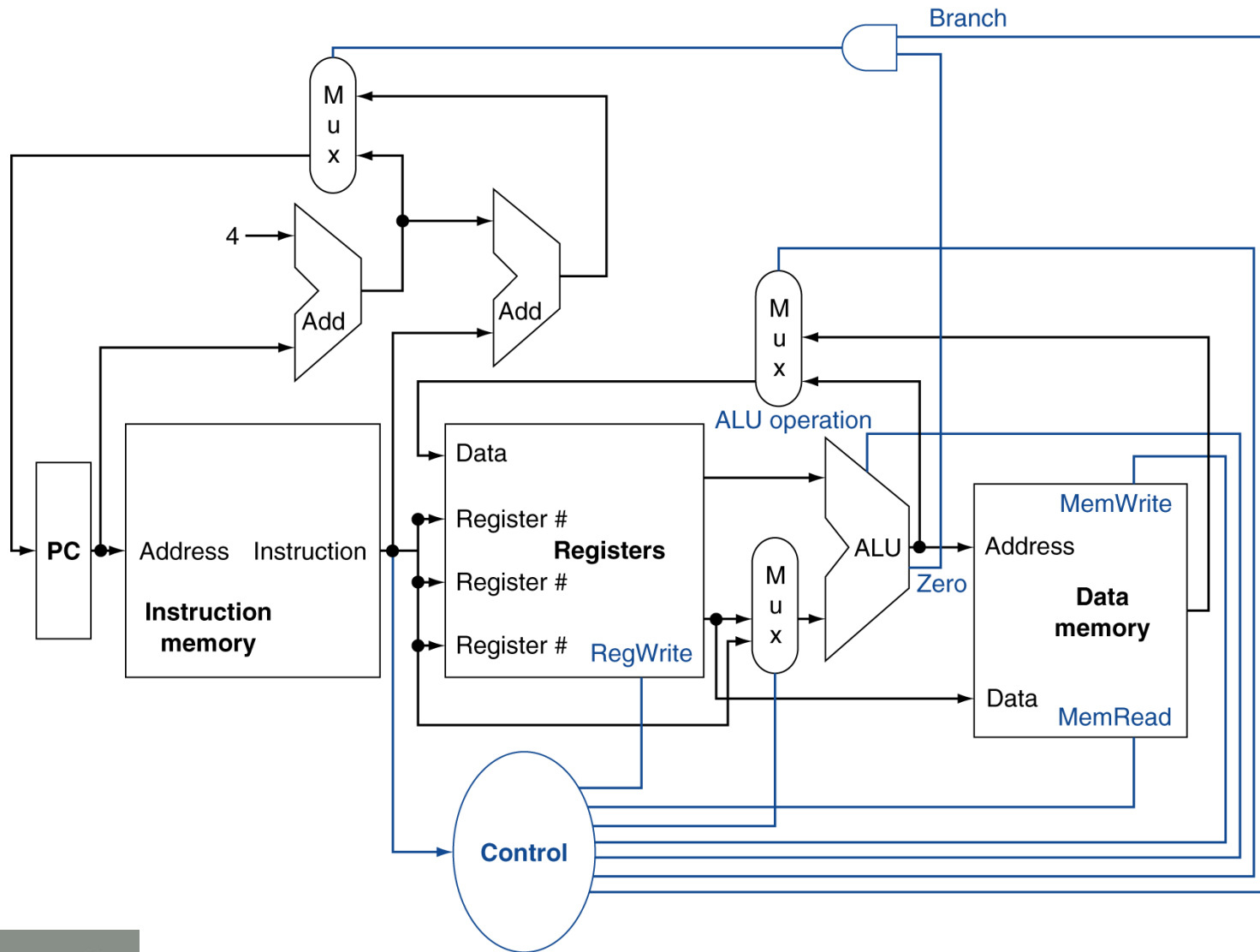


# Multiplexadores



- Os caminhos não podem se juntar diretamente
  - Uso de multiplexadores

# Controle



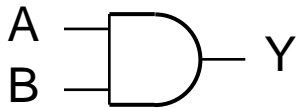
# Princípios do design lógico

- Informação sempre codificada em binário
  - Voltagem baixa = 0, Voltagem alta = 1
- Elementos combinacionais
  - Operam com os dados
  - A saída é uma função da entrada
- Elementos de estado (sequenciais)
  - Armazenam informação

# Elementos combinacionais

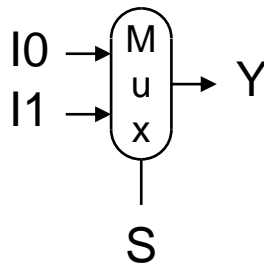
- Porta AND

- $Y = A \& B$



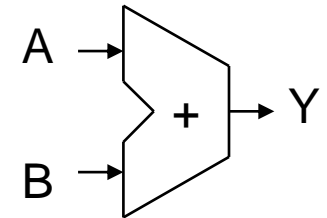
- Multiplexador

- $Y = S ? I1 : I0$



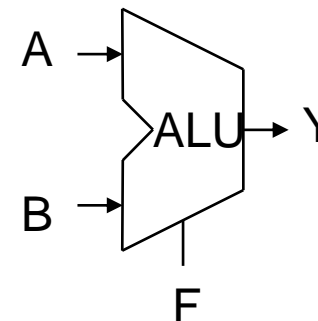
- Somador

- $Y = A + B$



- Unidade lógica e aritmética (ULA)

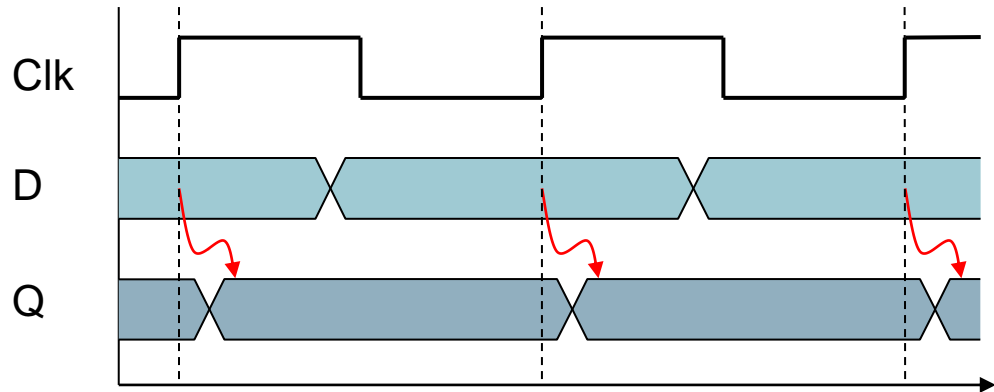
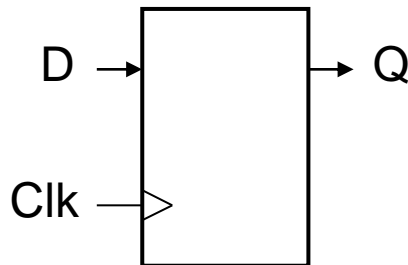
- $Y = F(A, B)$





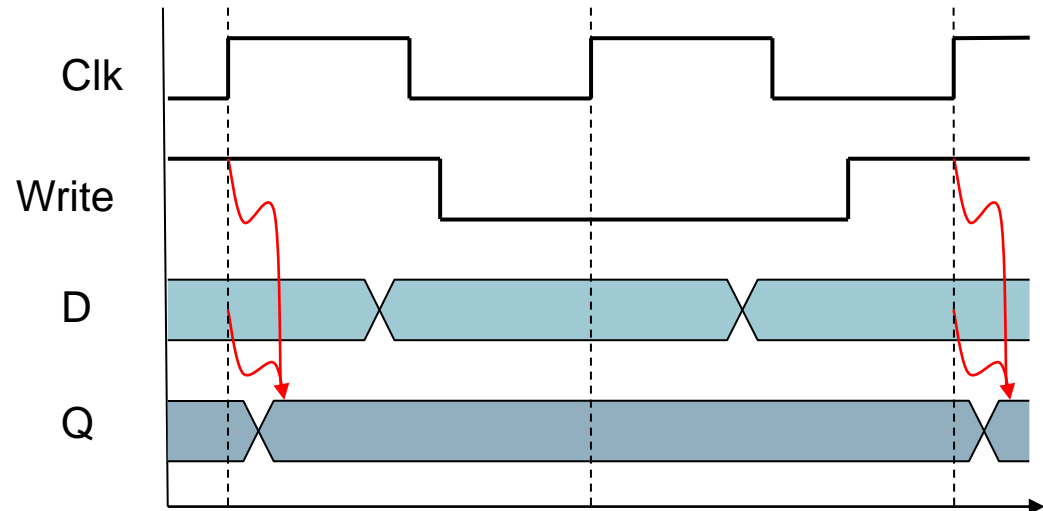
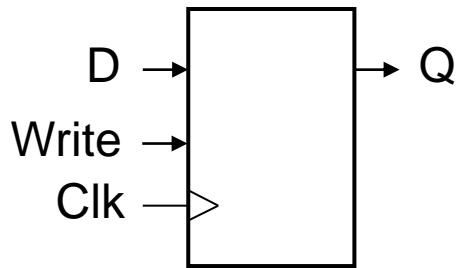
# Elementos sequenciais

- Registrador: armazena dados num circuito
  - Usa o sinal de clock para determinar quando atualizar o valor armazenado
  - Edge-triggered: atualiza quando Clk muda de 0 para 1



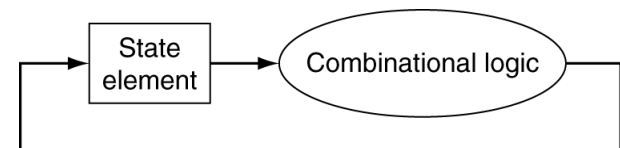
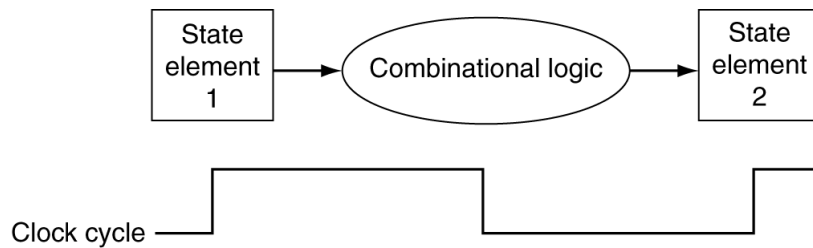
# Elementos sequenciais

- Registrador com controle de escrita
  - Apenas atualiza ao final do clock quando o controle de escrita for 1



# Metodologia de clock

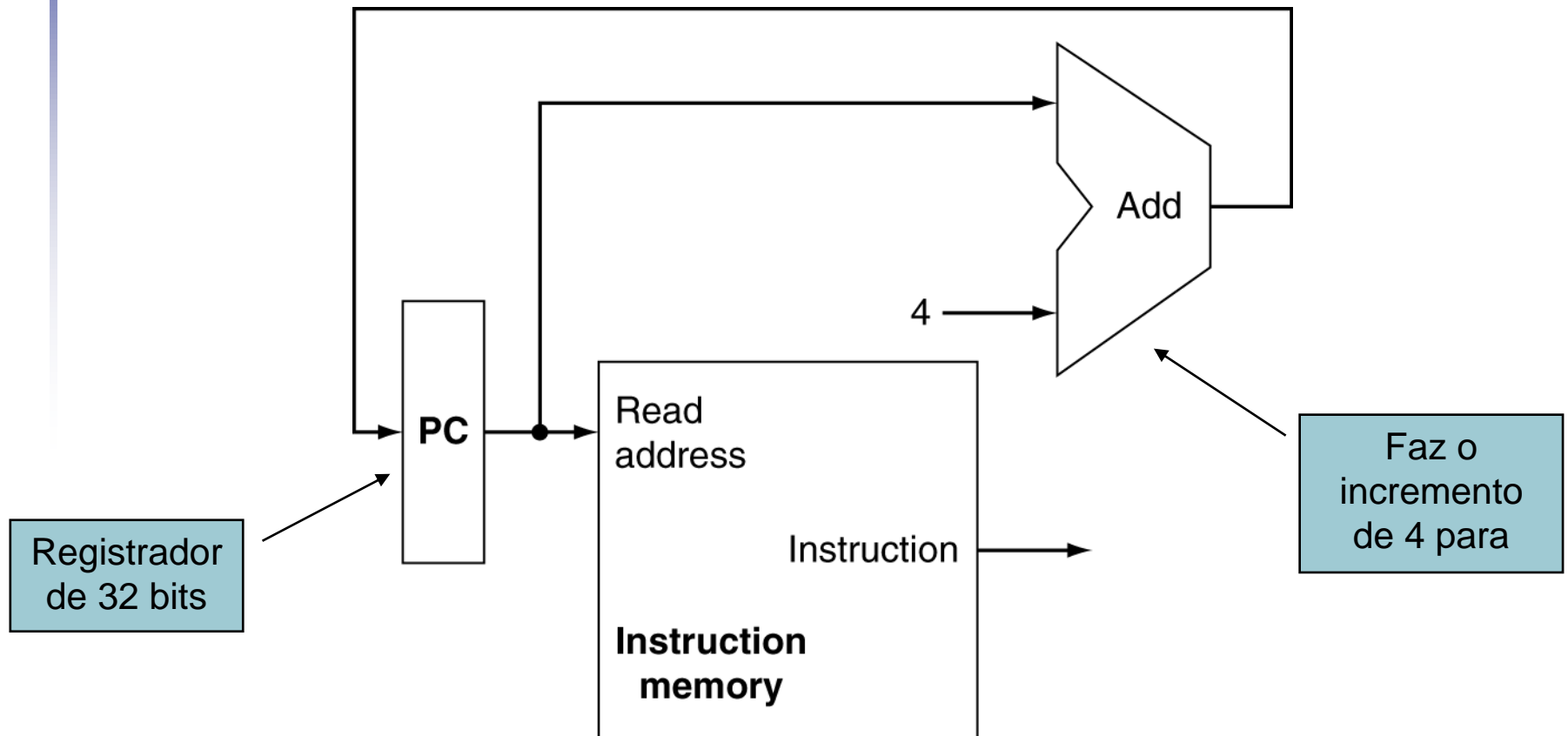
- A lógica combinacional transforma os dados durante um ciclo de clock
  - Entre os limites do clock
  - Entrada de um elemento de estado, saída para um elemento de estado
  - A instrução mais demorada determina a duração do clock



# Construindo um caminho de dados

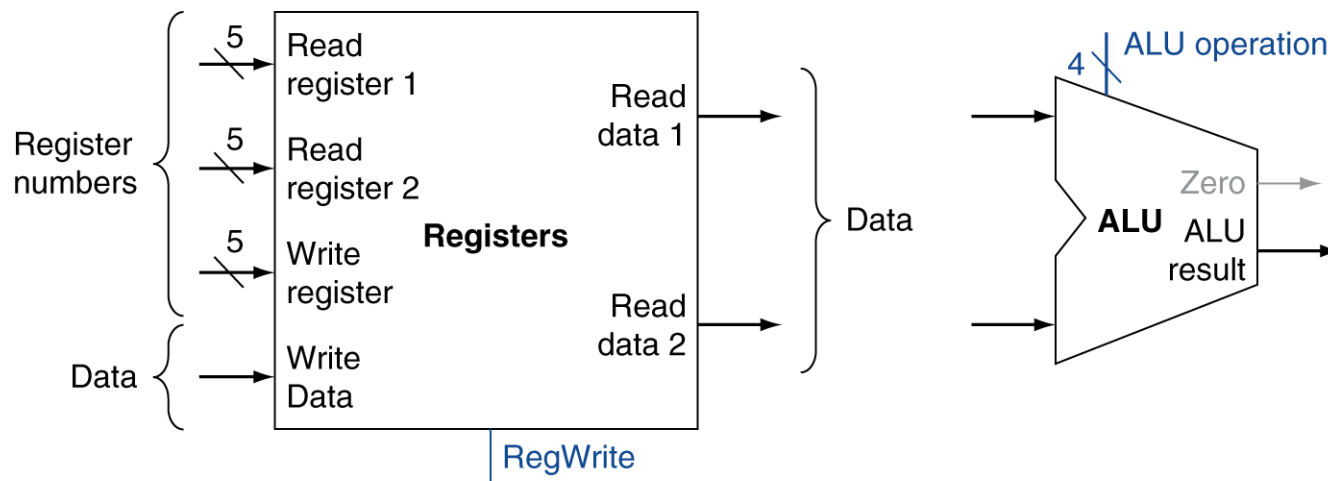
- Caminho de dados
  - Elementos que processam os dados e endereços numa CPU
    - Registradores, ULAs, multiplexadores, memórias, ....

# Fetch – obter a instrução



# Instruções do tipo R

- Lê dois registradores operandos
- Faz a operação lógica/aritmética
- Escreve o resultado num registrador

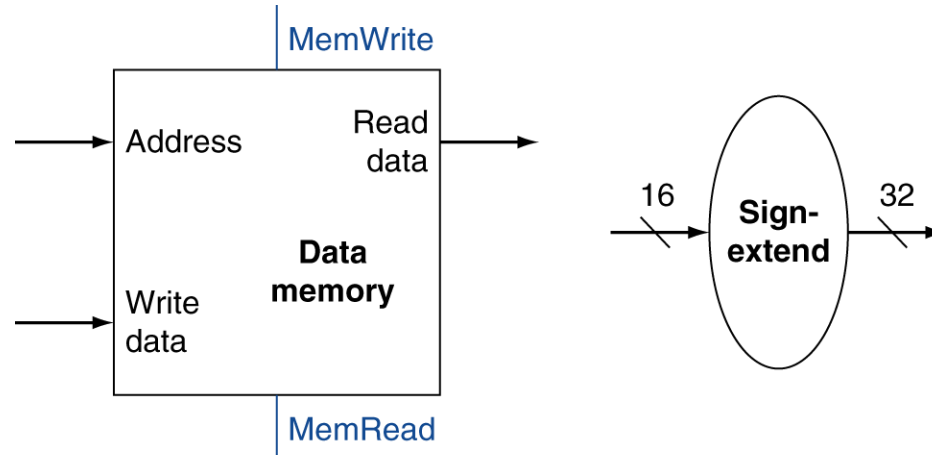
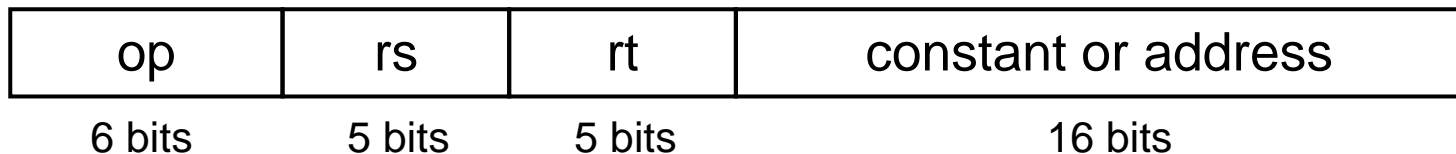


a. Registers

b. ALU

# Instruções load/store

- Lê os registradores operandos
- Calcula o endereço de memória usando o offset de 16 bits
  - Usa a ULA e também um extensor de sinal do offset
- Load: lê da memória e escreve no registrador
- Store: lê do registrador e escreve na memória



a. Data memory unit

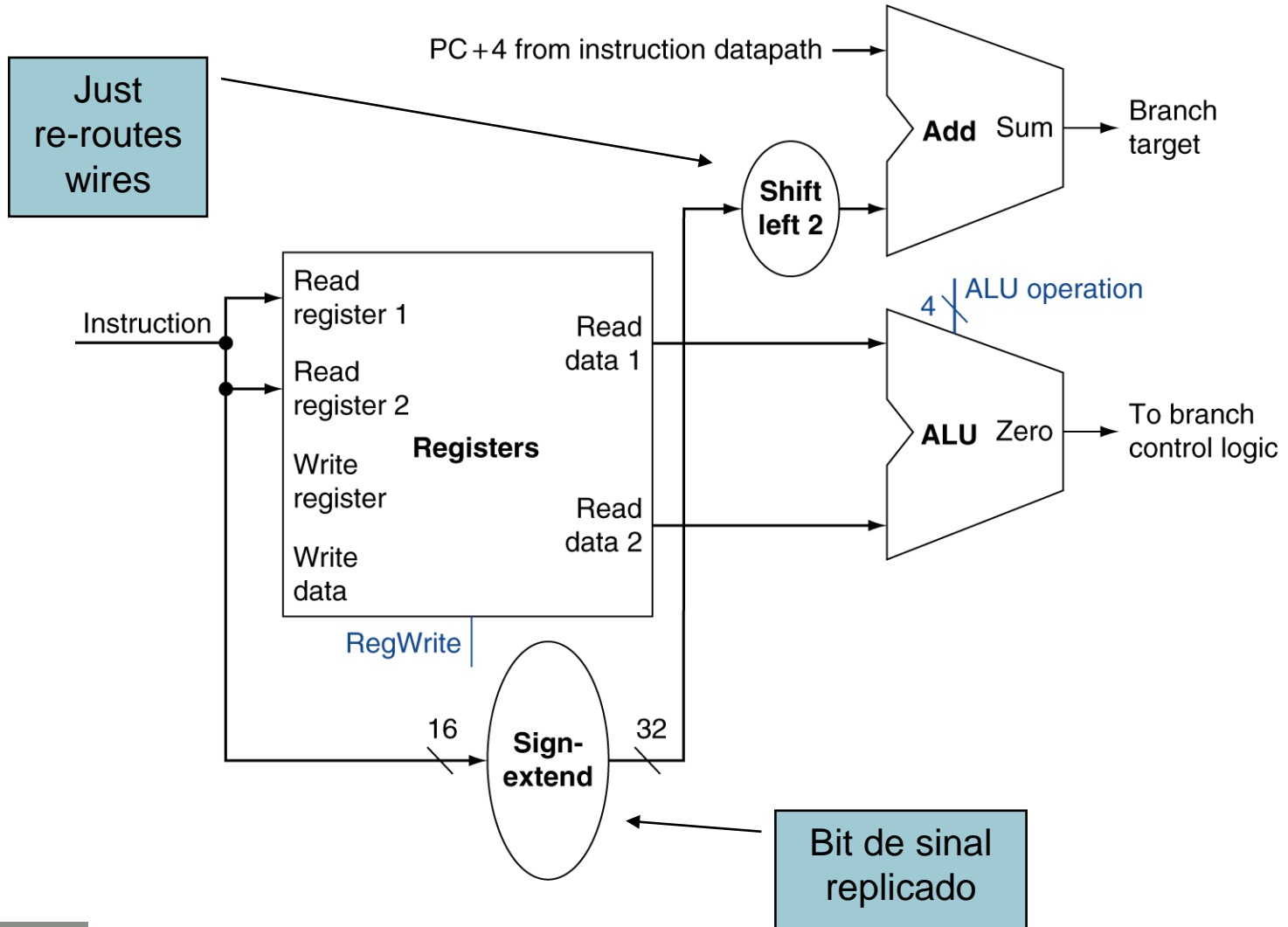
b. Sign extension unit

# Instruções de desvio

- Lê os registradores operandos
- Compara os operandos
  - Usa a ULA para subtrair e verifica se a saída é zero
- Calcula o endereço de destino do desvio
  - Extensão de sinal
  - 2 shifts à esquerda (palavra: 4 bytes)
  - Soma a PC + 4
    - PC + 4 já foi calculado no fetch da instrução



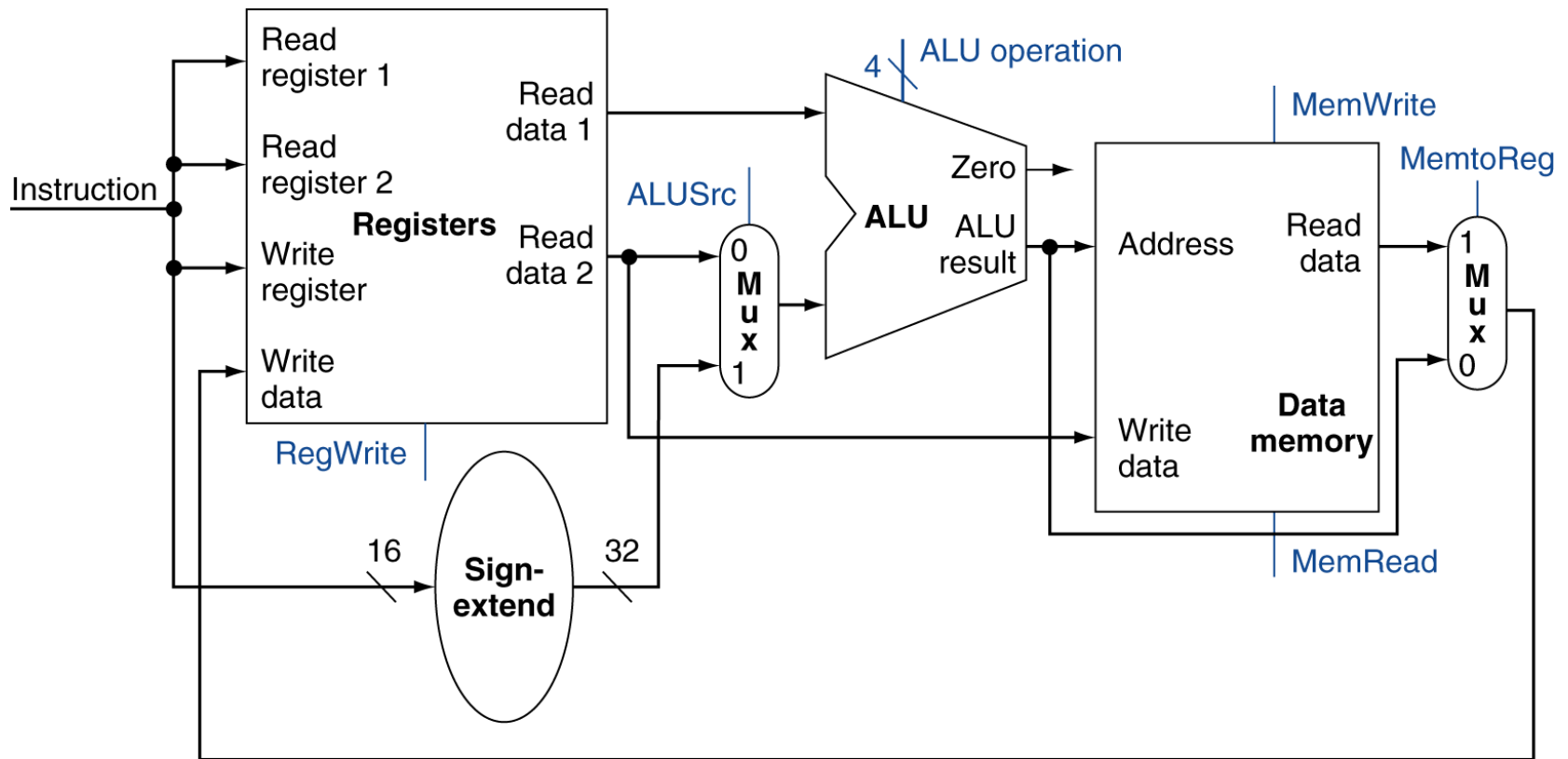
# Instruções de desvio



# Compondo os elementos

- O caminho de dados executa uma instrução em um único ciclo de clock
  - Cada elemento no caminho de dados pode fazer apenas uma coisa por vez
  - Portanto, há uma memória de dados e outra de instruções
- Usa multiplexadores onde é necessário alternar fontes de dados para diferentes instruções

# Caminho de dados composto



# Caminho de dados completo

