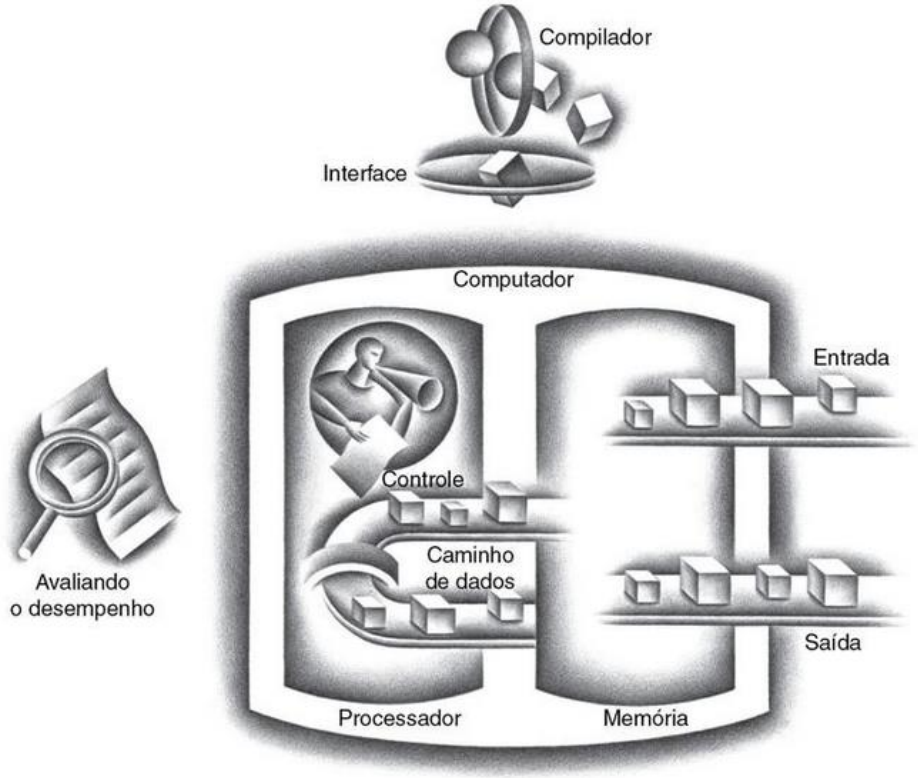
[Aula 07] Caminho de dados e controle 1

Prof. João F. Mari
joaof.mari@ufv.br

Roteiro

- Introdução
- Sinopse da implementação
- Visão abstrata da implementação do subconjunto MIPS mostrando as unidades funcionais principais e as conexões principais
- Implementação básica do subconjunto MIPS incluindo as linhas de controle e os multiplexadores necessários
- Métodos de temporização (*clocking*)
- CONSTRUINDO O CAMINHO DE DADOS


Compilador
Interface
Computador
Entrada
Controle
Caminho
de dados
Saída
Processador
Memória
Avaliando
o desempenho

INTRODUÇÃO

Introdução

- O desempenho de um programa depende:
  - Número de instruções (depende do conjunto de instruções)
  - Velocidade de clock (depende da implementação)
  - CPI - Número de ciclos gastos por instrução (depende da implementação do ISA)

- Implementação do MIPS simplificada:
  - Instruções de referência à memória: **`lw`**, **`sw`**
  - Instruções lógicas e aritméticas: **`add`**, **`sub`**, **`and`**, **`or`**, **`slt`**
  - Instruções de controle de fluxo (saltos): **`beq`**, **`j`**

- Implementação geral
  - BUSCA a instrução no endereço e memória apontado pelo contador de programa (PC). Atualiza o PC (PC=PC+4)
  - DECODIFICA a instrução. Lê os registradores
  - EXECUTA a instrução

- Todas as instruções (exceto j) usam a ULA após a leitura dos registradores.

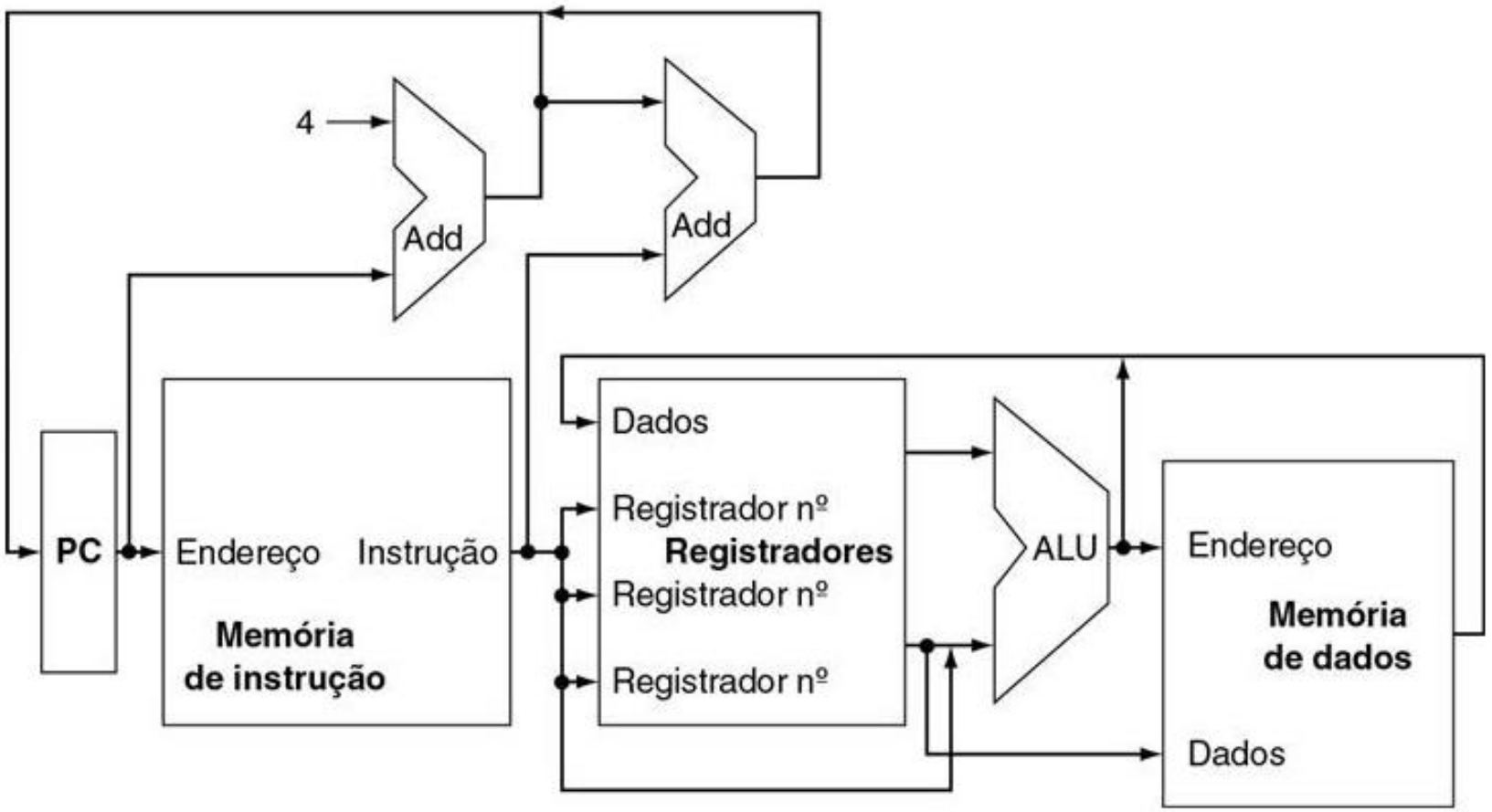
Introdução

- Implementação simplificada
  - Diretrizes:
    - Torne o caso comum mais rápido
- A simplicidade favorece a regularidade
  - As demais instruções podem ser implementadas com princípios semelhantes
  - A arquitetura do conjunto de instruções influencia os aspectos de implementação:
    - Instruções mais simples necessitam de uma implementação também mais simples
- Memória
  - Inicialmente as memórias de programa e controle separadas

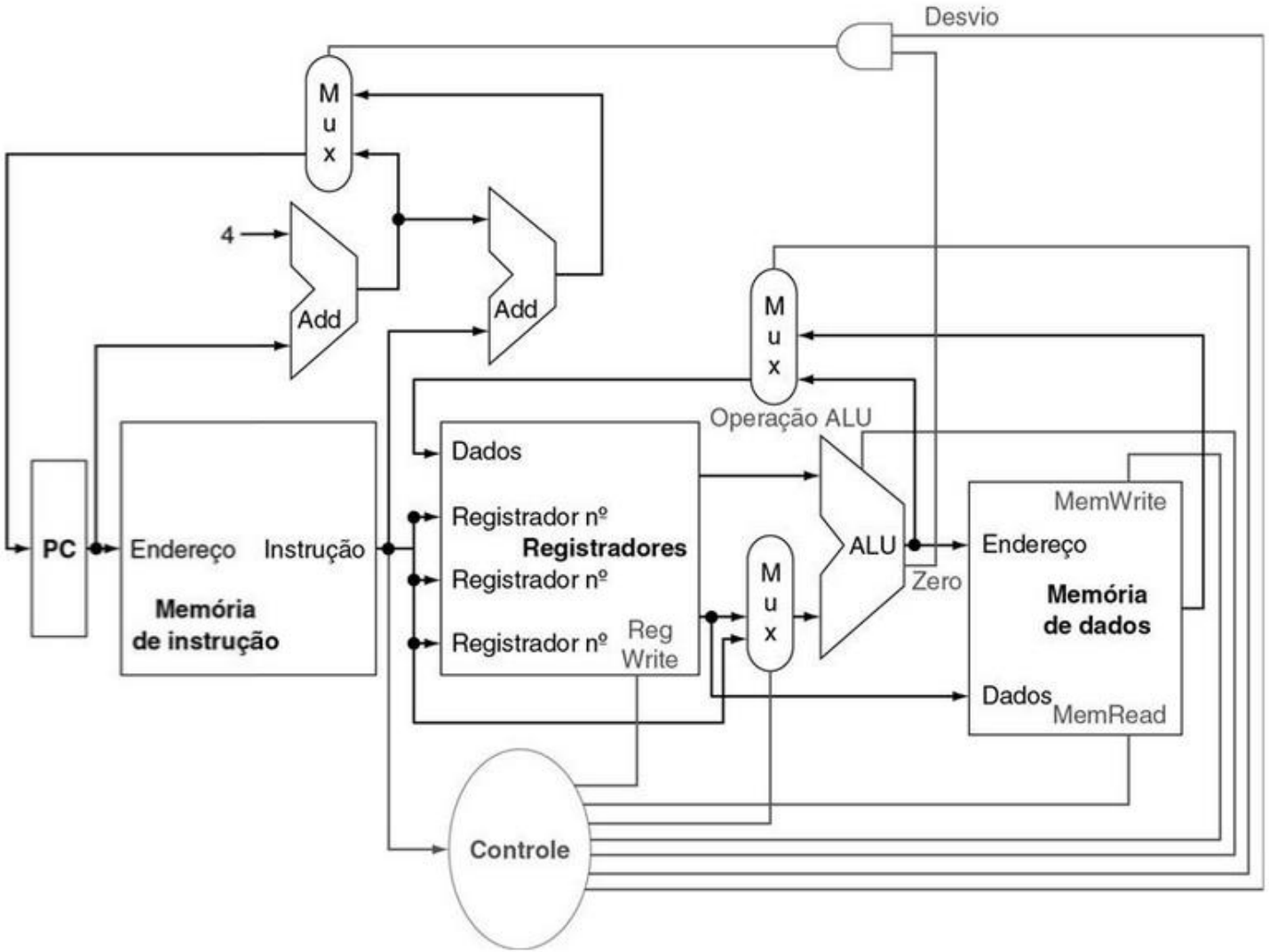
Sinopse da implementação

- Todas as instruções do conjunto de instruções têm os dois primeiros passos idênticos:
  - Enviar o valor armazenado no PC para a memória de programa e buscar a instrução dessa memória
  - Ler um ou dois registradores, usando os campos de instrução para selecionar os registradores a serem lidos.
  - Para a instrução load word, precisamos ler apenas um registrador, mas a maioria das outras instruções exige a leitura de dois registradores
- Mesmo entre diferentes classes de instruções, há algumas semelhanças:
  - Todas as classes utilizam a UAL após a leitura dos registradores:
  - Instruções de referência à memória: efetuar o cálculo do endereço
    - Instruções aritméticas e lógicas: efetuar a operação
    - Desvios condicionais: efetuar comparação (subtração)
- Após usar a UAL, as ações necessárias diferem
  - Referência a memória: escreve dado na memória
  - Instrução aritmética: escreve dado no registrador

Visão abstrata da implementação do subconjunto MIPS mostrando as unidades funcionais principais e as conexões principais

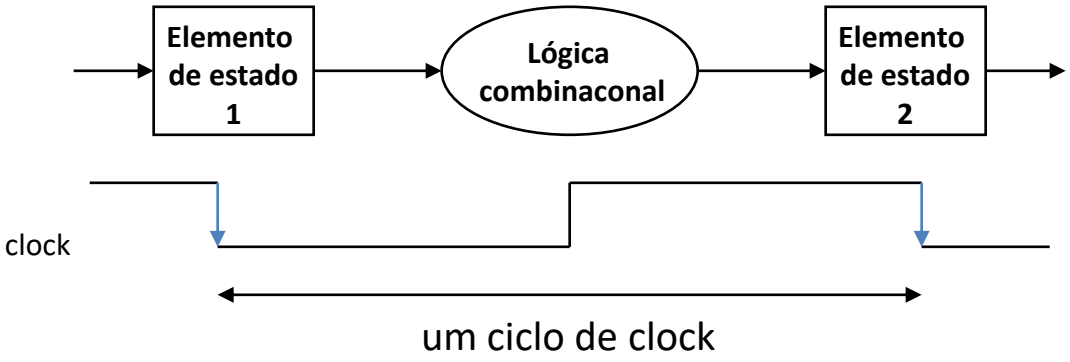


Implementação básica do subconjunto MIPS incluindo as linhas de controle e os multiplexadores necessários



Métodos de temporização (*clocking*)

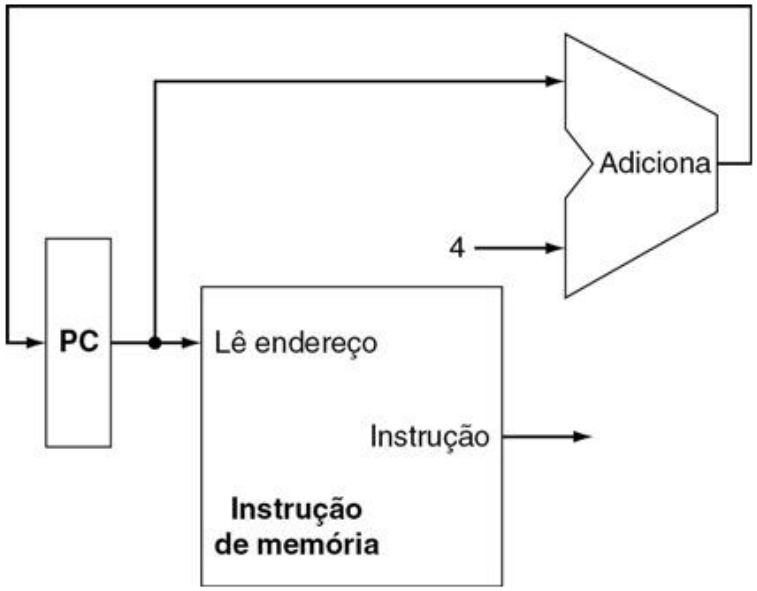
- Define quando os sinais podem ser lidos e quando podem ser escritos
  - Evita circunstâncias onde um sinal é lido ao mesmo tempo que o mesmo sinal foi escrito;
    - A leitura pode retornar o valor antigo, o valor recente ou uma combinação entre os dois
- Sincronização acionada por transição
  - Significa que quaisquer valores armazenados em um elemento lógico sequencial são atualizados apenas em uma transição de clock
  - Permite que um elemento de estado seja lido e escrito no mesmo ciclo de clock
    - Não cria disputa que poderia levar a valores de dados indeterminados.
  - **O período de clock necessita ser longo o suficiente para que os valores de saída estabilizem.**
- Apenas os elementos de estado podem armazenar valores de dados,
  - Qualquer coleção de lógica combinatória precisa ter suas entradas vindo de um conjunto de elementos de estado
  - Suas saídas são escritas em um conjunto de elementos de estado.



CONSTRUINDO O CAMINHO DE DADOS

Busca de instruções

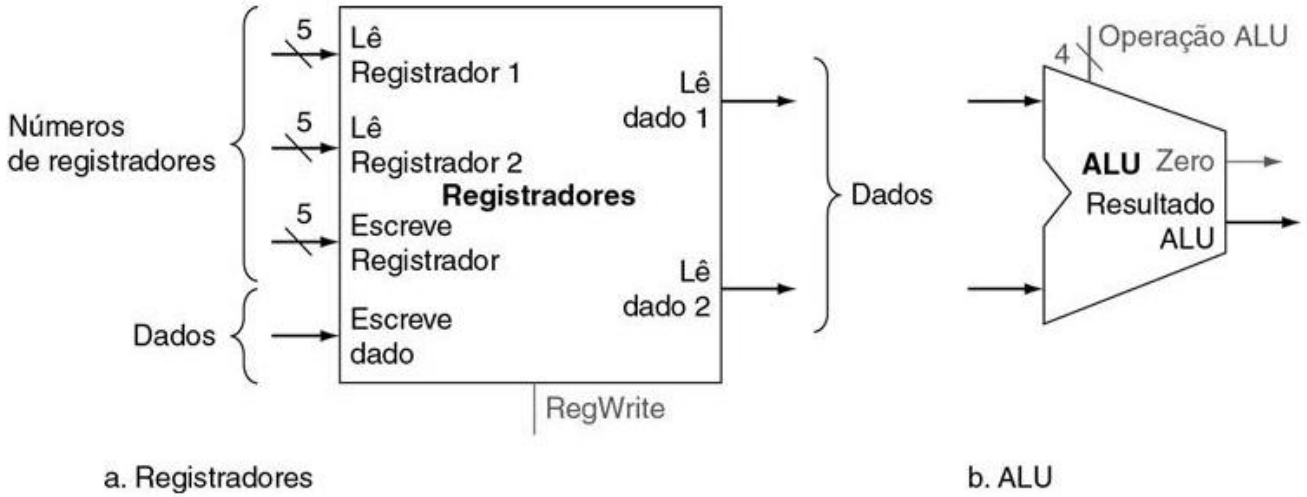
- Lê a instrução da memória de instruções
  - Atualiza o PC para guardar o endereço da próxima instrução



- PC é atualizado a cada, por isso não precisa de um sinal de controle de escrita.
- A memória de instruções é lida a cada ciclo, por isso não precisa de um sinal de controle de leitura.

Banco de registradores

- Instruções de formato R possuem três operandos de registrador
  - Lê duas palavras de dados do banco de registradores e escrever uma palavra de dados no banco de registradores.
  - 3 entradas de 5 bits (32 registradores):
    - 2 entradas com endereço dos registradores lidos
    - 1 entrada com endereço do registrador escrito
- 2 saídas de 32 bits: operandos para a UAL
- 1 entrada de 32 bits: escrita do resultado
  - As escritas são controladas pelo sinal de controle de escrita, que precisa estar ativo para que um escrita ocorra na transição do clock.



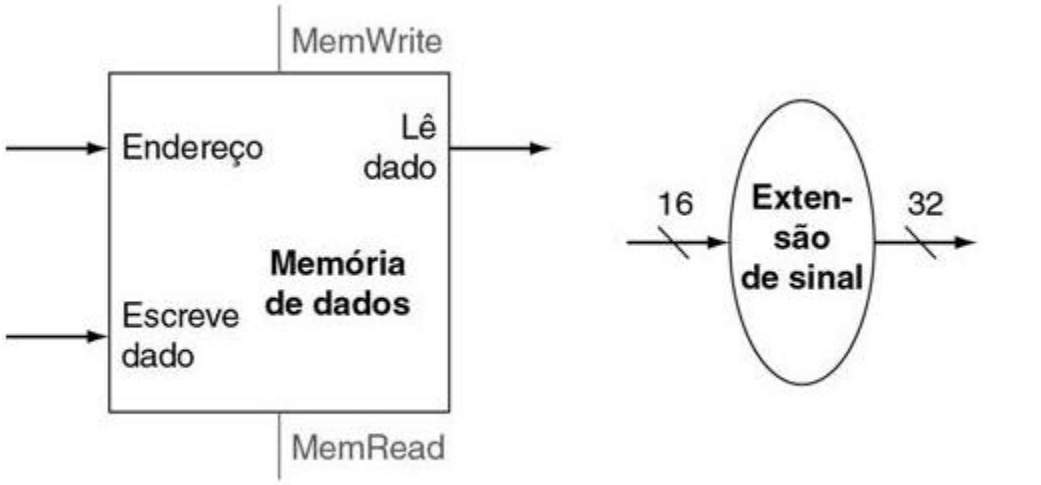
Instruções do formato R

- Executa a operação (op and funct) utilizando como operandos os valores em rs e rt
- Armazenam o resultado no banco de registradores (no endereço rd)
- Todas as instruções do tipo R precisam
  - Ler dois registradores
  - Realizar uma operação na UAL com os conteúdos dos registradores
  - Escrever o resultado em um registrador
- Instruções aritméticas lógicas: ADD, SUB, AND, OR
  - Ex: add $t1, $t2, $t3

Tipo R	opcode	rs	rt	rd	shamt	funct
	31-26	25-19	20-16	15-11	10-6	5-0

Instruções de load e store

- `lw $t1, offset($t2)` e `sw $t1, offset($t2)`
- Endereço de memória é calculado somando o registrador base ($t2 no exemplo) ao número de 16 bits sem sinal estendido
  - `sw`: o valor a ser armazenado na memória de dados é lido do registrador.
  - `lw`: o valor é lido da memória de dados e escrito no registrador
- É necessária uma unidade para estender o sinal de 16 para 32 bits e uma memória para ler e escrever os dados.

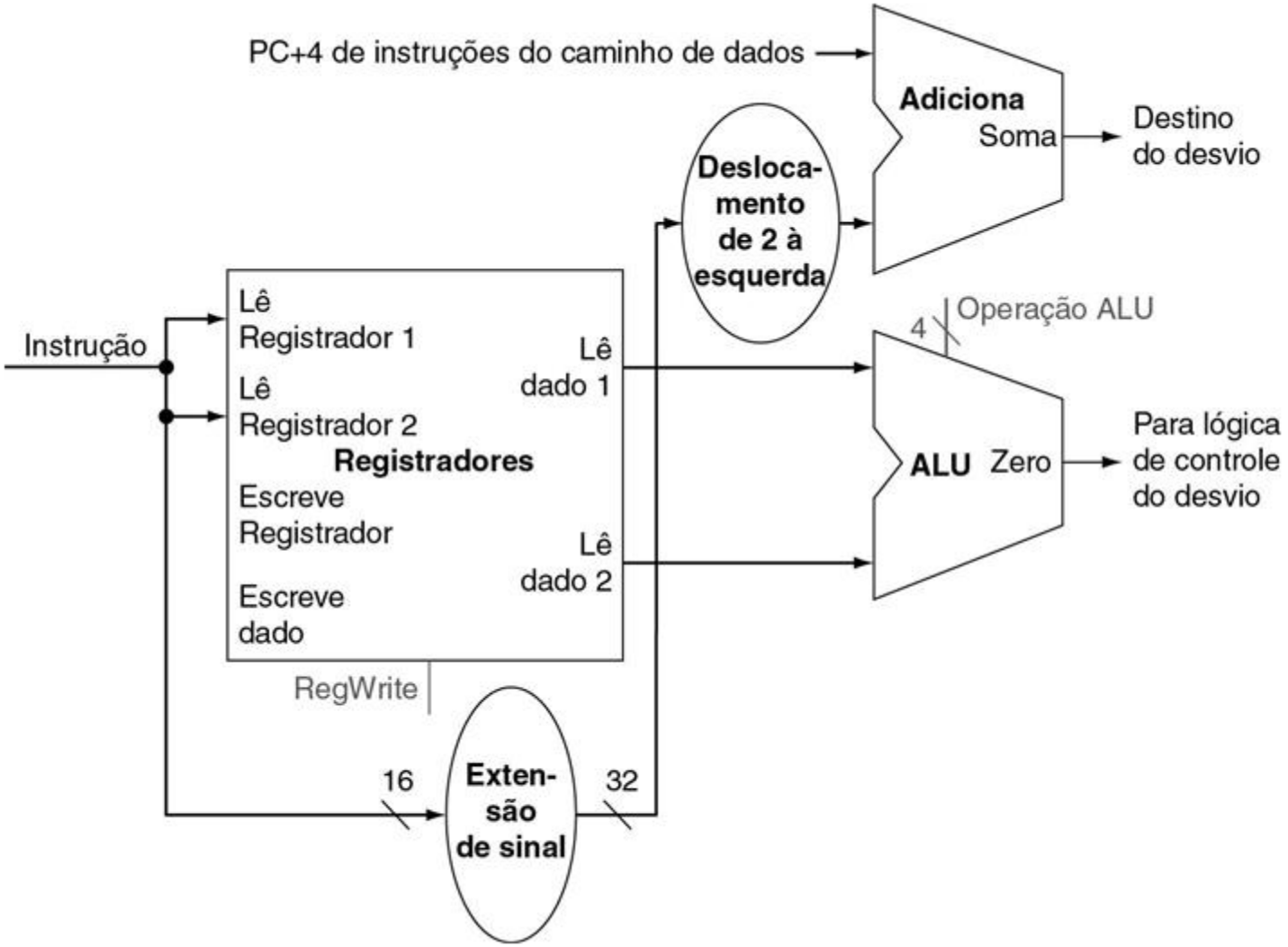


a. Unidade de memória de dados b. Unidade de extensão de sinal

A instrução beq

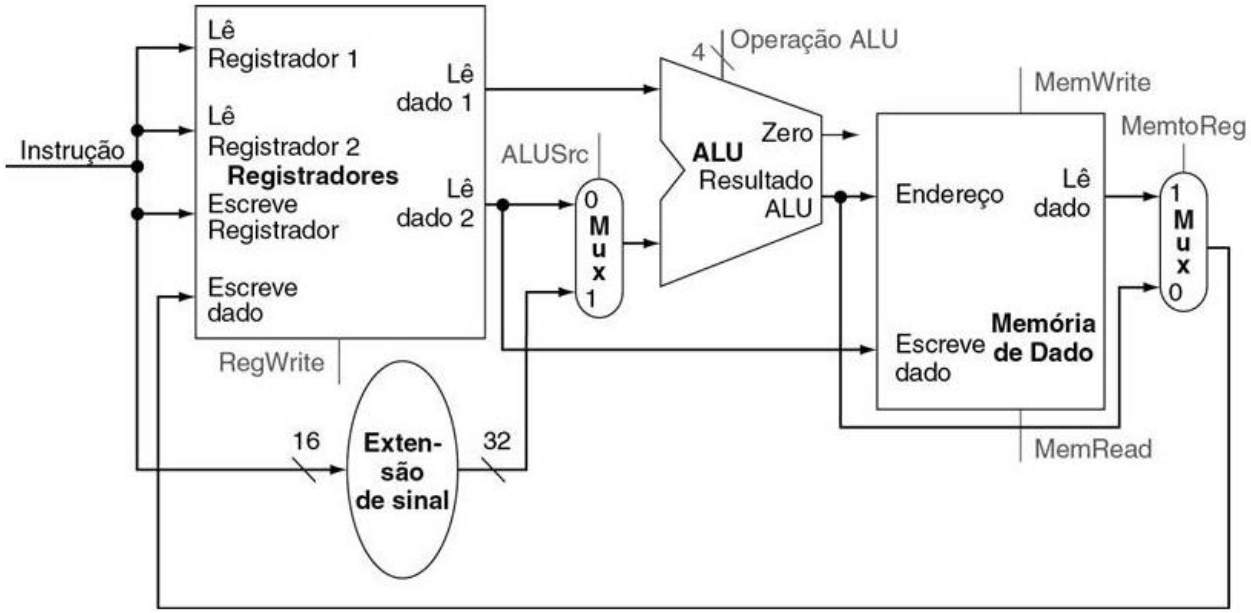
- Possui três operandos
  - Dois registradores utilizados para a comparação
  - Offset indicando o endereço de memória para o deslocamento (PC + offset)
- Instruções de desvio
  - A base para o cálculo do endereço de desvio é o endereço da instrução seguinte ao desvio.
    - Como calculamos PC + 4 no caminho de dados para a busca de instruções, é fácil usar esse valor como a base para calcular o endereço de destino do desvio.
  - O campo offset é deslocado 2 bits para a esquerda de modo que resulte em uma offset de uma palavra;
    - Aumenta a amplitude do salto por um fator igual a 4
- Além de calcular o endereço do desvio, é necessário verificar se o desvio deve ser executado ou não, de acordo com a comparação entre os dois registradores
- Assim, o caminho de dados do desvio precisa de duas operações:
  - Calcular o endereço de destino do desvio
  - Comparar o conteúdo do registrador (sinal zero da UAL)

A instrução beq



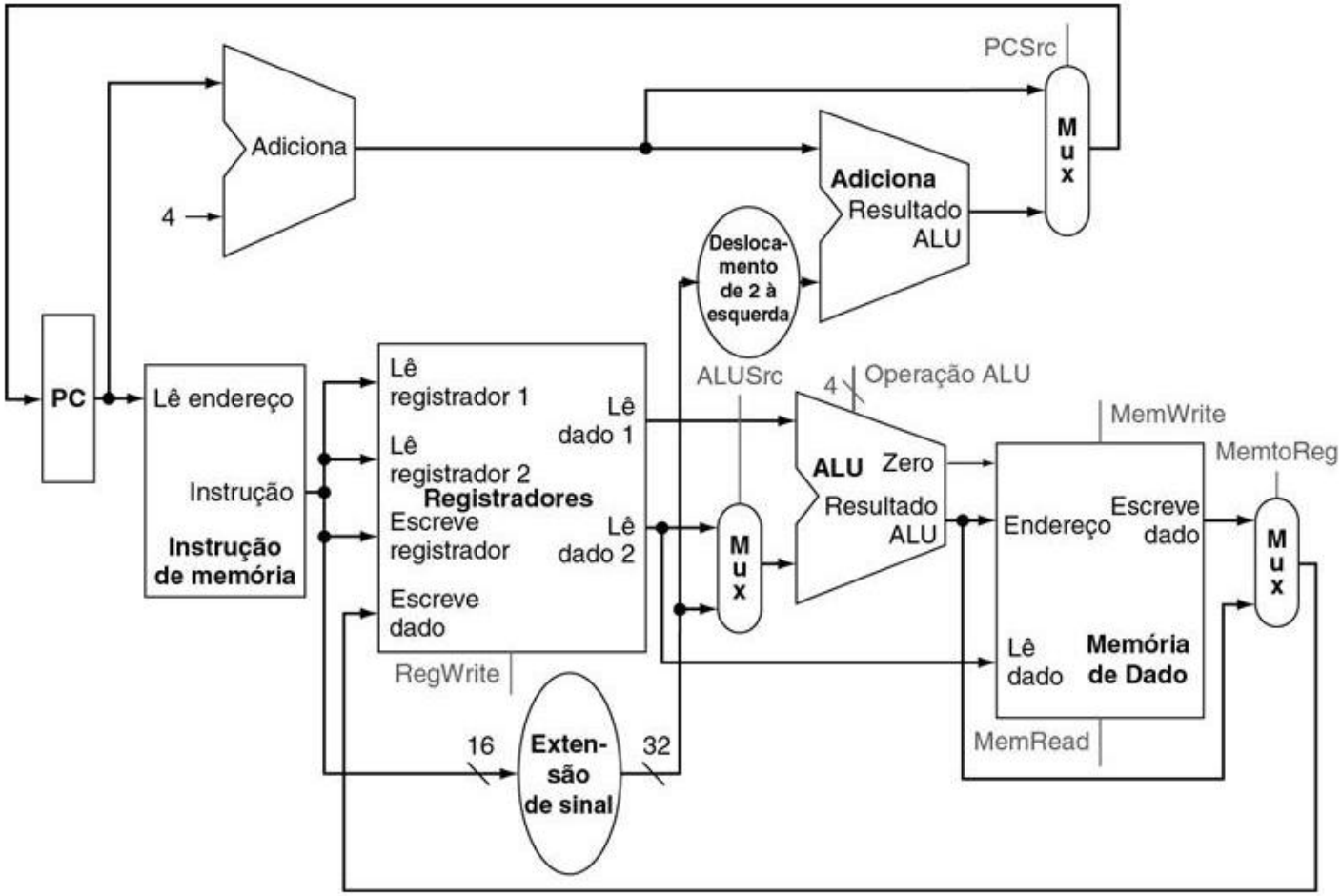
Operações lógicas e de acesso à memória

- As instruções lógicas e aritméticas usam a ALU com as entradas vindas de dois registradores;
- As instruções de acesso à memória também podem usar a ALU para fazer o cálculo do endereço
  - Nesse caso, a segunda entrada é o campo offset de 16 bits com o sinal estendido da instrução
- O valor armazenado em um registrador de destino vem da ALU (para um instrução do tipo R) ou da memória (para um load)



Integrando os caminhos de dados

- Os caminhos de dados são unidos e linhas de controle são adicionadas, assim como os multiplexadores necessários



Bibliografia

1. PATTERSON, D.A; HENNESSY, J.L. **Organização e Projeto de Computadores: A Interface Hardware/Software**. 3a. Ed. Elsevier, 2005.
   - Capítulo ***5***.

2. Notas de aula do prof. Luciano J. Senger:
   - http://www.ljsenger.net/classroom.html

3. Notas de aula da Profa. Mary Jane Irwin
   - CSE 331 Computer Organization and Design
   - http://www.cse.psu.edu/research/mdl/mji/mjicourses

FIM

- FIM:
  - **Aula 07 –** Caminho de dados e controle 1
- Próxima aula:
  - **Aula 08** – Caminho de dados e controle 2 – MIPS Monociclo