Controle e processamento

Arquitetura de Computadores

Bruno Prado

Departamento de Computação / UFS

Introdução

- Todo sistema computacional pode ser logicamente dividido em duas partes fundamentais

Introdução

- Todo sistema computacional pode ser logicamente dividido em duas partes fundamentais
  - Controle
    - Controla as unidades de processamento
    - Organiza a sequência de operações

Introdução

- Todo sistema computacional pode ser logicamente dividido em duas partes fundamentais
  - Controle
    - Controla as unidades de processamento
    - Organiza a sequência de operações
  - Processamento
    - Realiza a execução das operações
    - Utiliza os caminhos de dados

Introdução

- Projeto de lógica de operação
  - Assíncrono ou combinacional
    - Geram saídas a partir de entradas
    - O tempo de resposta depende dos componentes envolvidos e do valor de entrada utilizado

Introdução

- Projeto de lógica de operação
  - Assíncrono ou combinacional
    - Geram saídas a partir de entradas
    - O tempo de resposta depende dos componentes envolvidos e do valor de entrada utilizado

O desempenho é limitado pela maior latência de chaveamento na sequência de componentes

Introdução

- Projeto de lógica de operação
  - Síncrono ou baseado em relógio
    - Geram saídas baseadas em ciclos de relógio
    - A frequência máxima de operação depende do projeto e da tecnologia utilizada na implementação

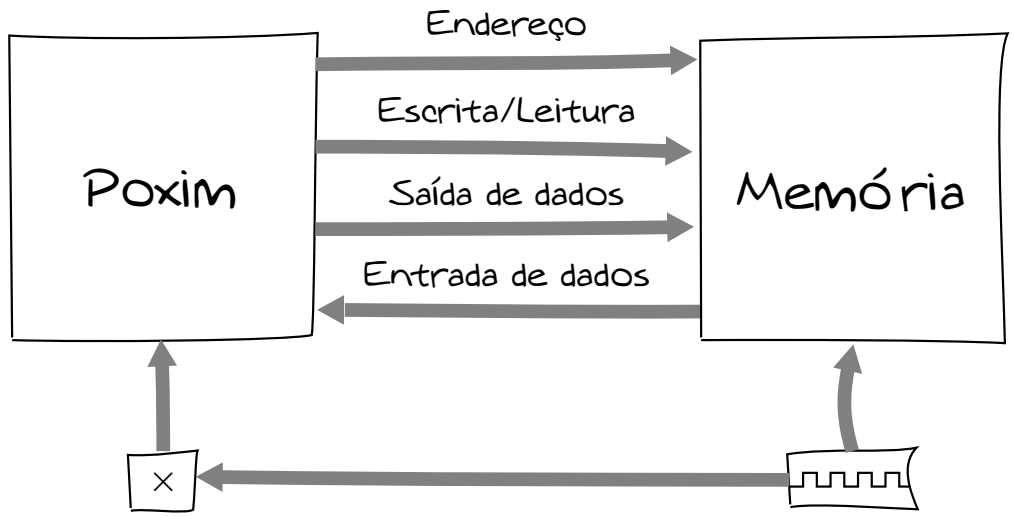
Introdução

- Projeto de lógica de operação
  - Síncrono ou baseado em relógio
    - Geram saídas baseadas em ciclos de relógio
    - A frequência máxima de operação depende do projeto e da tecnologia utilizada na implementação

O desempenho depende somente da frequência de operação e da taxa de execução (operações por ciclo)

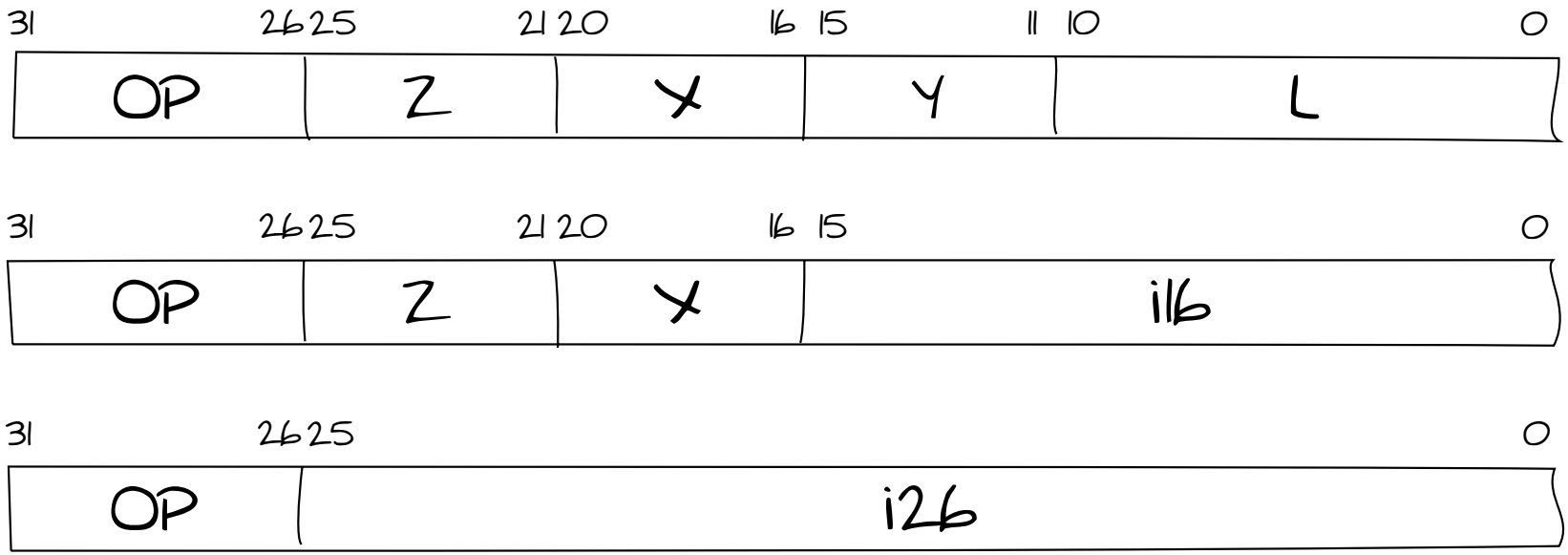
Visão geral

- Arquitetura Poxim
  - Processador síncrono de 32 bits
  - Memória Von Neumann



Visão geral

- Formato de instruções
  - 6 bits para operação (OP)
  - 5 bits para operandos (Z, X, Y)
  - 11 bits para uso livre (L)
  - 16 bits para imediato (I16)
  - 26 bits para imediato (I26)



Unidade de controle

- Papel do controle em um processador
  - Coordena todos os componentes do sistema, definindo a ordem e temporização dos eventos
  - Cada instrução é convertida em uma sequência de operações em uma máquina de estados que fará o controle das unidades de processamento

Unidade de controle

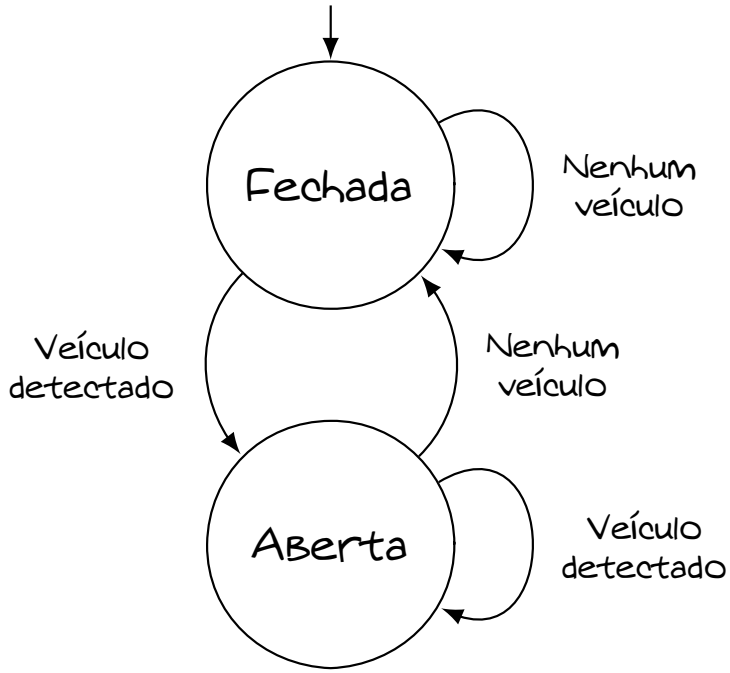
- Papel do controle em um processador
  - Coordena todos os componentes do sistema, definindo a ordem e temporização dos eventos
  - Cada instrução é convertida em uma sequência de operações em uma máquina de estados que fará o controle das unidades de processamento
- Abordagens
  - *Hardwired*
    - Implementação fixa baseada nos componentes
    - Funcionamento rápido

Unidade de controle

- Papel do controle em um processador
  - Coordena todos os componentes do sistema, definindo a ordem e temporização dos eventos
  - Cada instrução é convertida em uma sequência de operações em uma máquina de estados que fará o controle das unidades de processamento
- Abordagens
  - *Hardwired*
    - Implementação fixa baseada nos componentes
    - Funcionamento rápido
  - Micro-programada
    - Controle por micro-instruções reprogramáveis
    - Flexibilidade na modificação do projeto de controle

Unidade de controle

- Máquina de estados finita (FSM)
  - É definida pelos estados inicial, final ou de aceitação e por sua função de transição de estado



- Controle de entrada de estacionamento
  - Estados de cancela aberta e fechada (estado inicial)
  - Funções de transição: Veículo detectado $\rightarrow$ Cancela aberta e Nenhum veículo $\rightarrow$ Cancela fechada

Unidade de controle

- Definições de máquina de estados finita
  - *Mealy*: o estado e as entradas atuais definem os valores de saída da máquina (assíncrona)
  - *Moore*: somente pelo estado atual são definidas quais saídas serão geradas (síncrona)

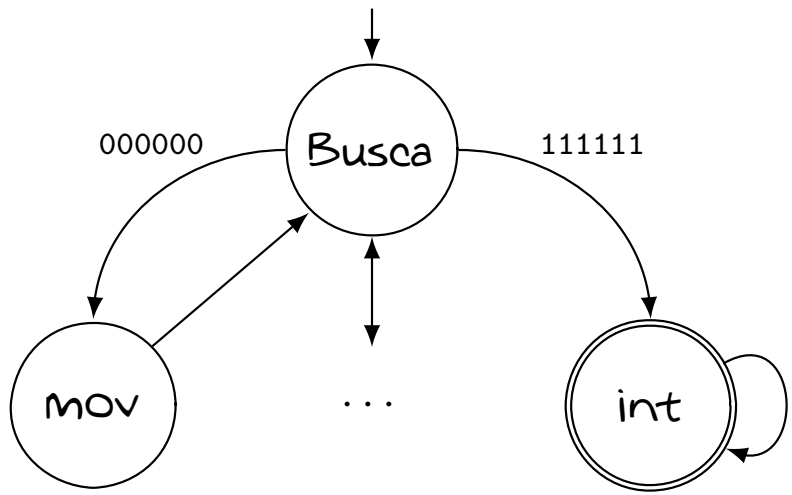
Unidade de controle

- Definições de máquina de estados finita
  - *Mealy*: o estado e as entradas atuais definem os valores de saída da máquina (assíncrona)
  - *Moore*: somente pelo estado atual são definidas quais saídas serão geradas (síncrona)

Ambas as definições podem ser utilizadas em conjunto, podendo utilizar múltiplas entradas e saídas

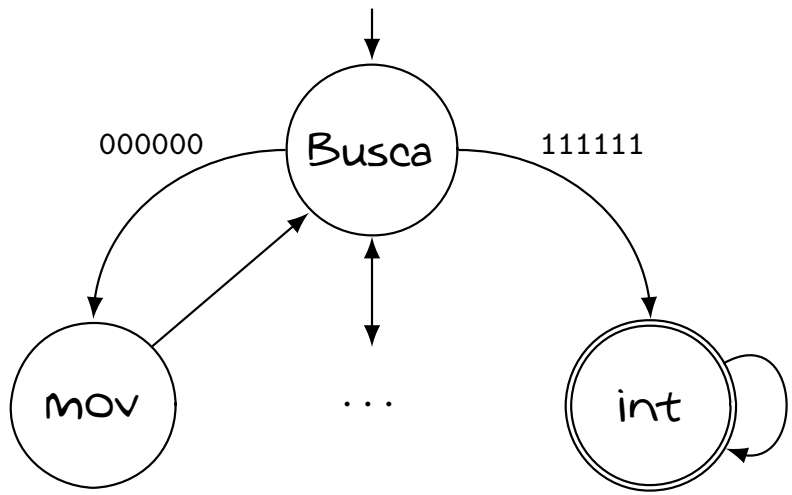
Unidade de controle

- Máquina de estados da unidade de controle
  - Estado inicial (inicialização ou *reset*)
  - Funções de transição (código de operação)
  - Estado final (interrupção de término)



Unidade de controle

- Máquina de estados da unidade de controle
  - Estado inicial (inicialização ou *reset*)
  - Funções de transição (código de operação)
  - Estado final (interrupção de término)



Cada estado possui a sequência de controle para executar sua respectiva operação

Unidade de controle

- O processador executa as instruções em um ciclo infinito de busca-decodificação-execução
  1. Buscar instrução armazenada na memória
  2. Decodificar a operação da instrução
  3. Obter os operandos necessários
  4. Executar o comportamento
  5. Armazenar os resultados
  6. Voltar para o passo 1

Buscar instrução

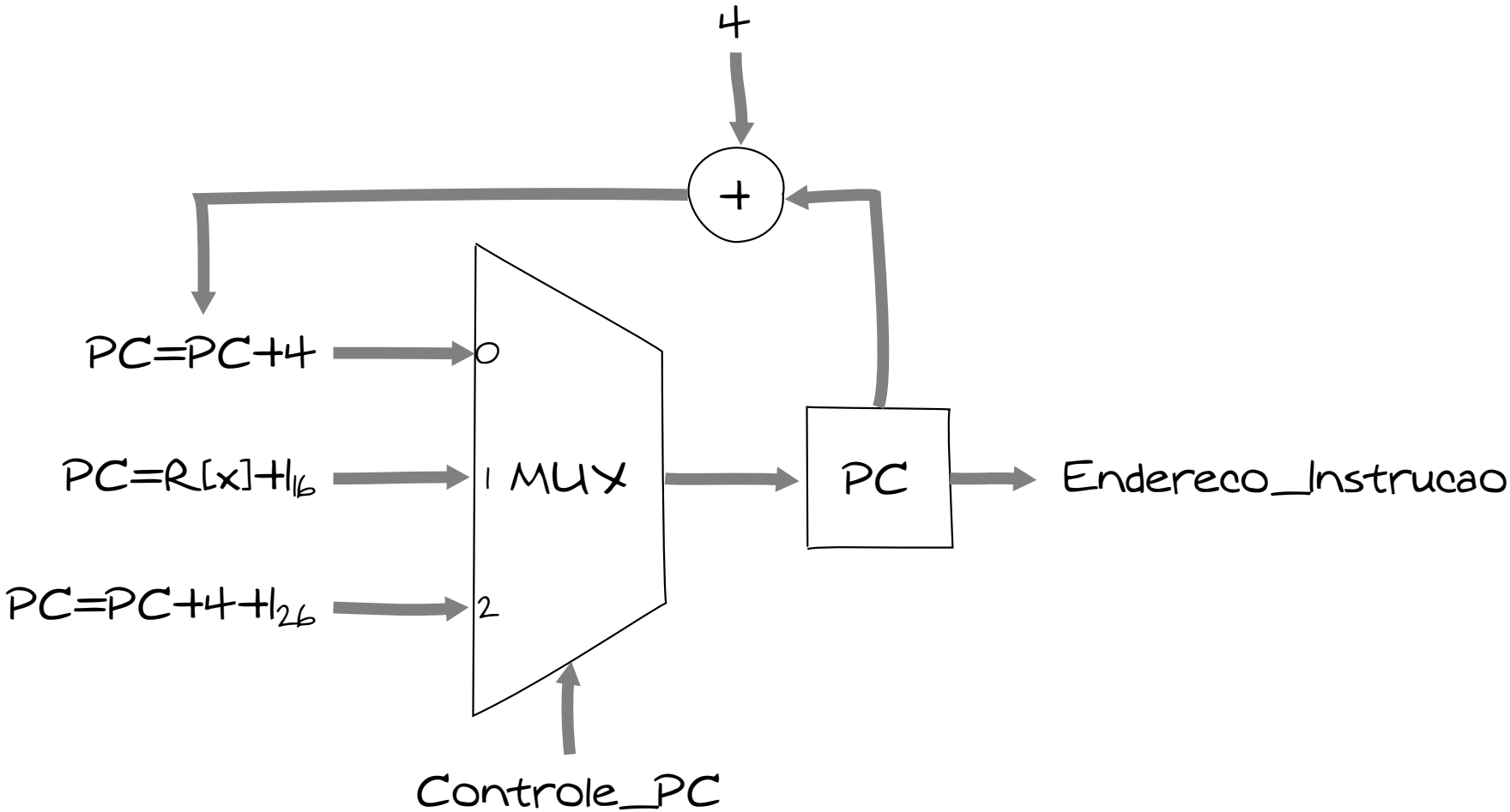
- Buscar instrução armazenada na memória
  - Cálculo do contador de programa (PC)
    - Sequencial: $PC = PC + 4$
    - Desvio absoluto: $PC = R[x] + I16$
    - Desvio relativo: $PC = PC + 4 + I26$

Buscar instrução

- Buscar instrução armazenada na memória
  - Cálculo do contador de programa (*PC*)
    - Sequencial: $PC = PC + 4$
    - Desvio absoluto: $PC = R[x] + I16$
    - Desvio relativo: $PC = PC + 4 + I26$
  - Endereçamento da memória ($Instrução = MEM[PC]$)
    - O endereço da memória recebe o valor do *PC*
    - A memória é configurada para leitura (0)
    - Após uma quantidade determinada de ciclos, a instrução é obtida da memória

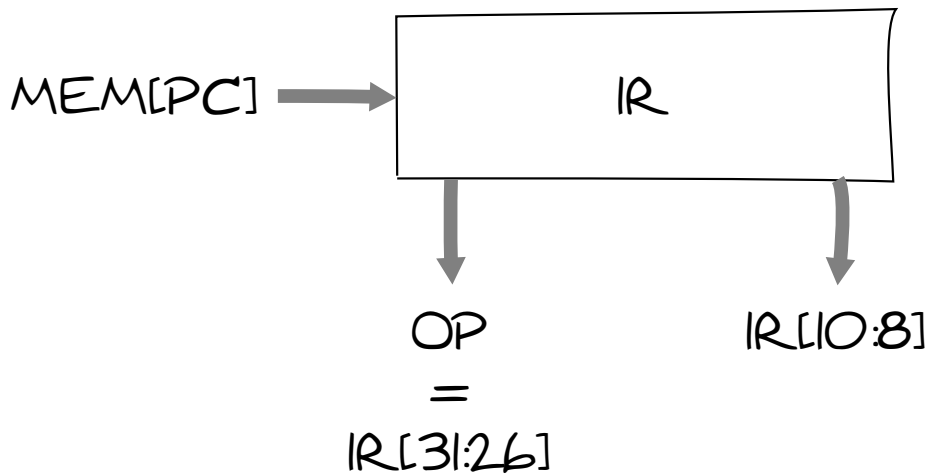
Buscar instrução

- Buscar instrução armazenada na memória
  - Diagrama de blocos



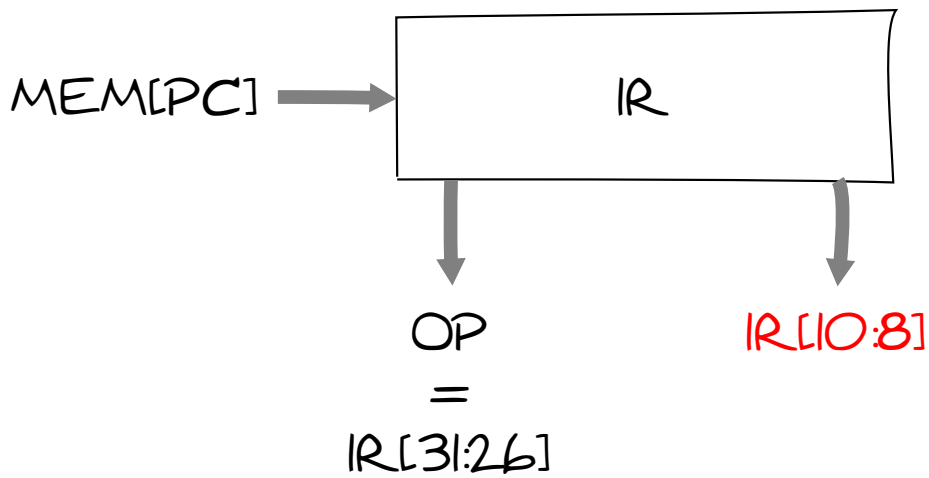
Decodificar instrução

- Decodificar a operação da instrução
  - A instrução está armazenada no registrador de instrução (IR) que será usado para decodificação da operação pela unidade de controle



Decodificar instrução

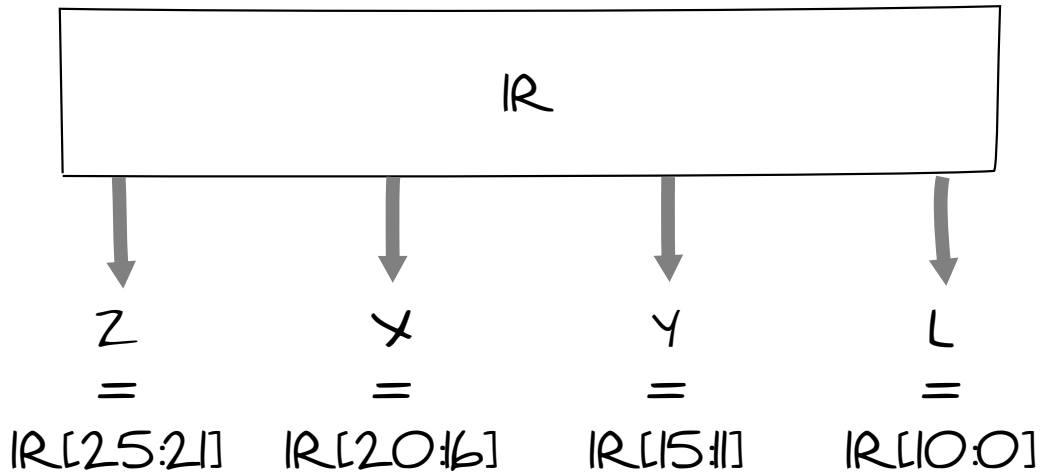
- Decodificar a operação da instrução
  - A instrução está armazenada no registrador de instrução (IR) que será usado para decodificação da operação pela unidade de controle



Regularidade $\longleftrightarrow$ Simplicidade

Buscar operandos

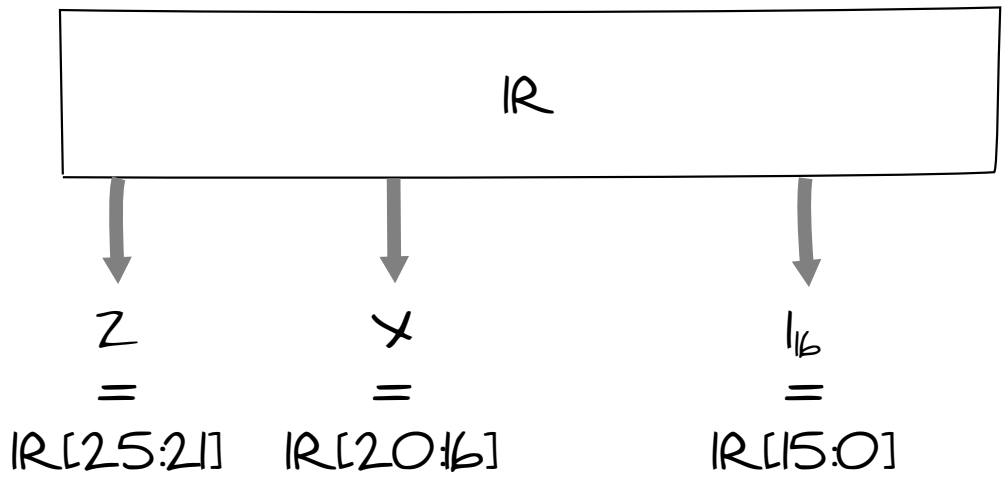
- Obter os operandos necessários
  - Tipo U (Z, X, Y e L)



  - Indexação de registradores: Z, X, Y, $L_{4:0}$ e $L_{10:6}$
  - Campo auxiliar de decodificação: $L_{10:8}$

Buscar operandos

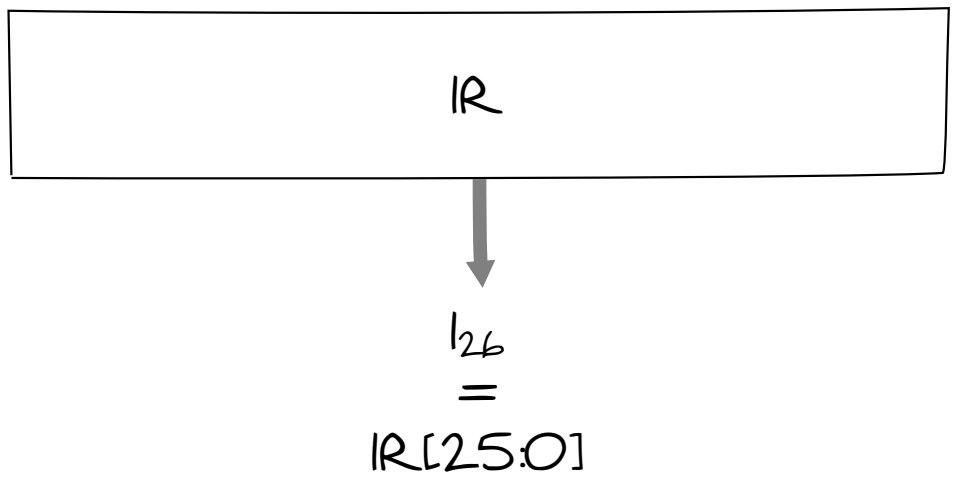
- Obter os operandos necessários
  - Tipo F (Z, X, I16)



  - Indexação de registradores: *Z* e *X*
  - Valor imediato: *I16*

Buscar operandos

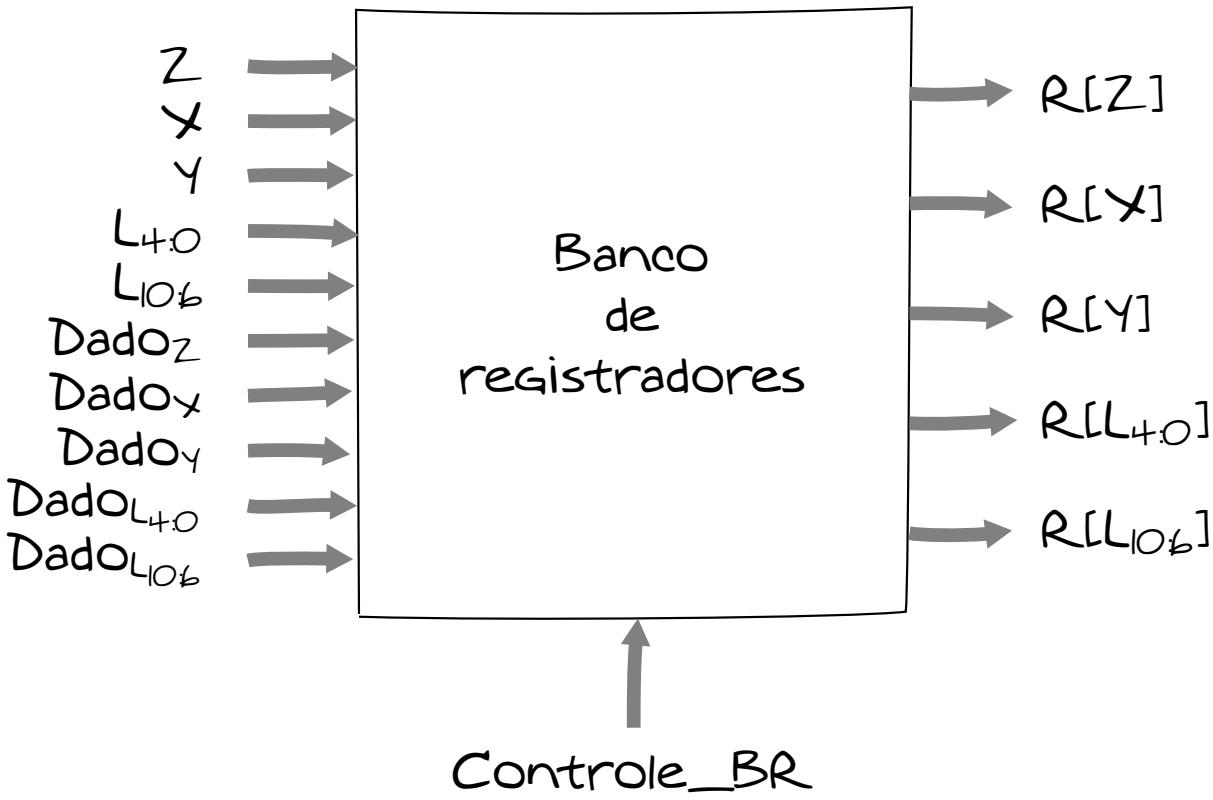
- Obter os operandos necessários
  - Tipo S (I26)



  - Valor imediato: *I*26

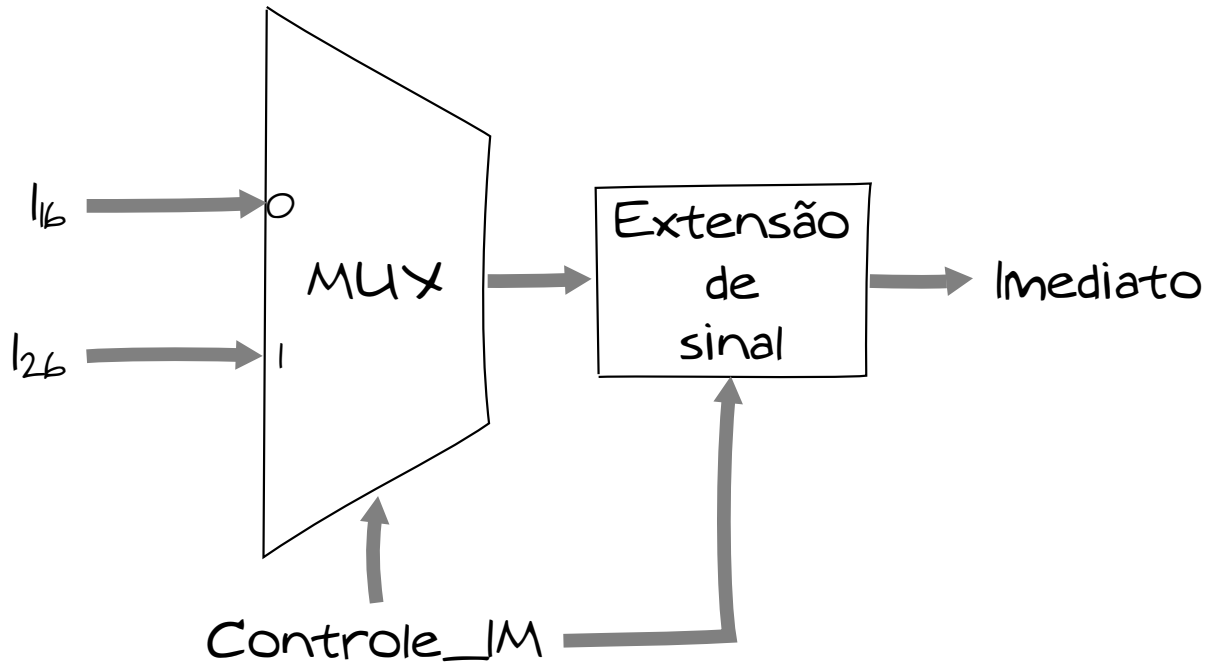
Buscar operandos

- Obter os operandos necessários
  - Acesso ao banco de registradores



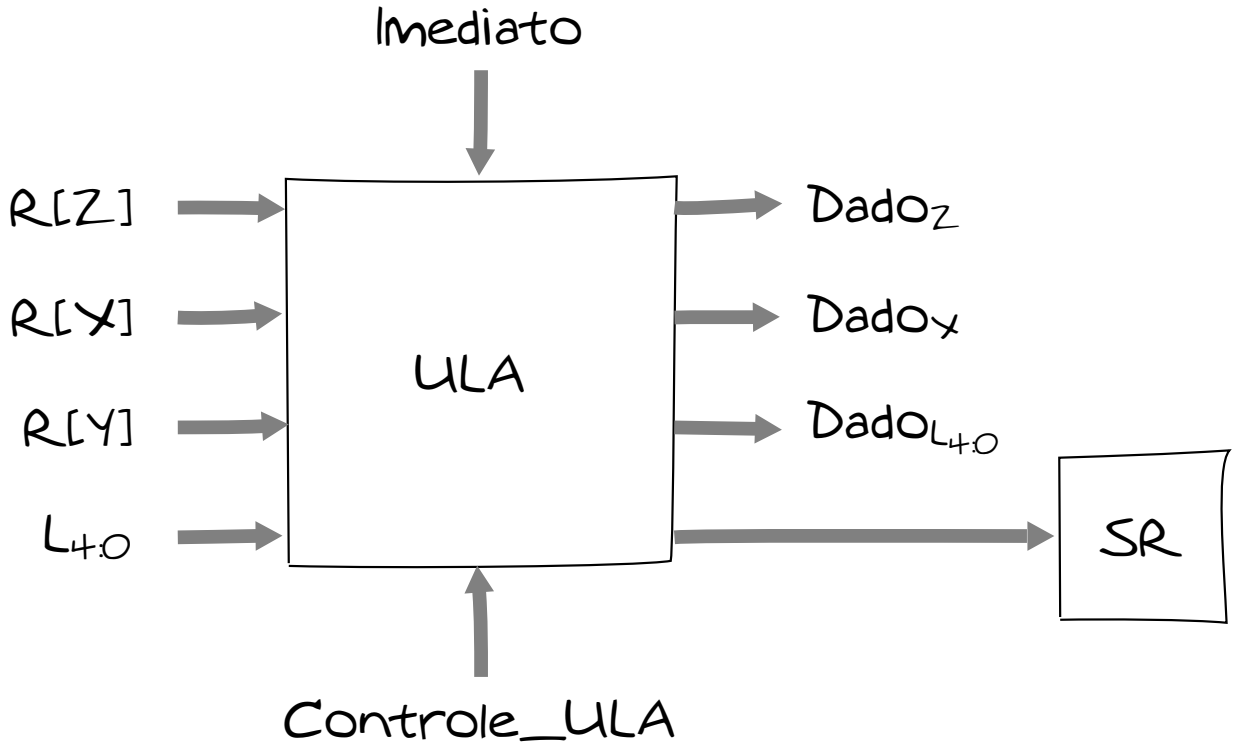
Buscar operandos

- Obter os operandos necessários
  - Extensão de sinal dos valores imediatos



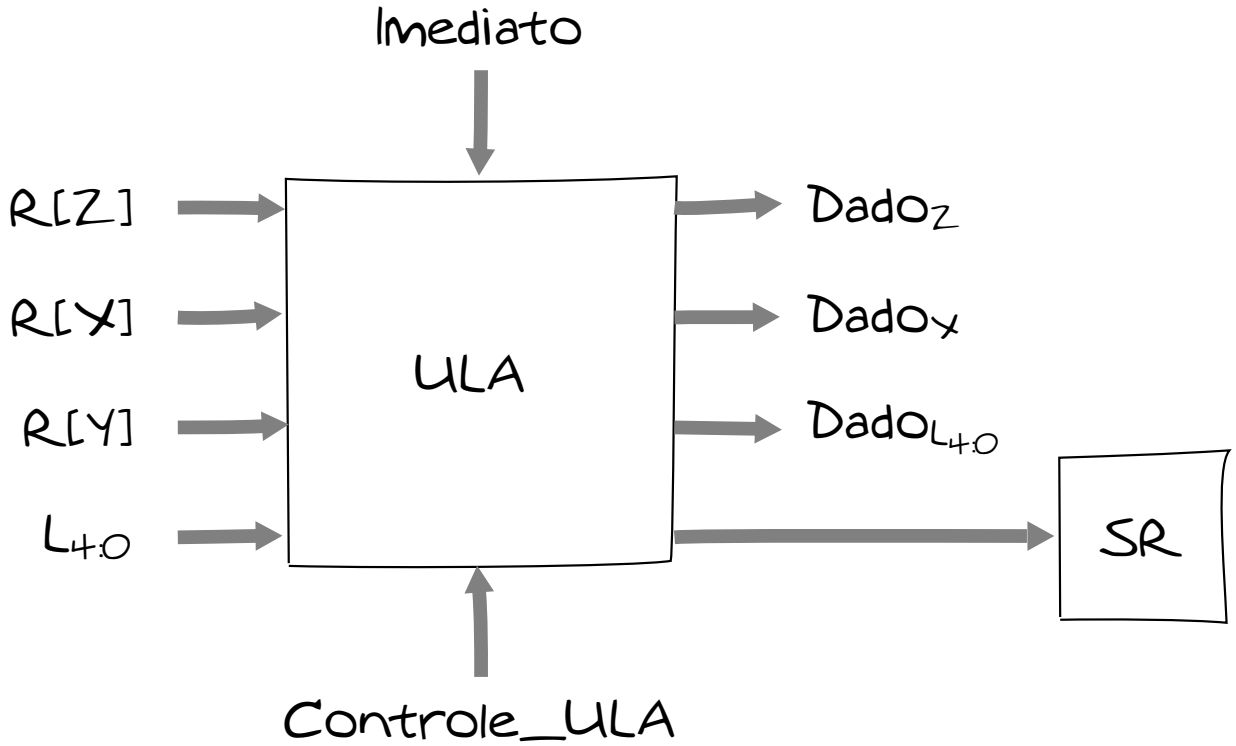
Executar operação

- Executar o comportamento
  - Unidade de Lógica e Aritmética (ULA)



Executar operação

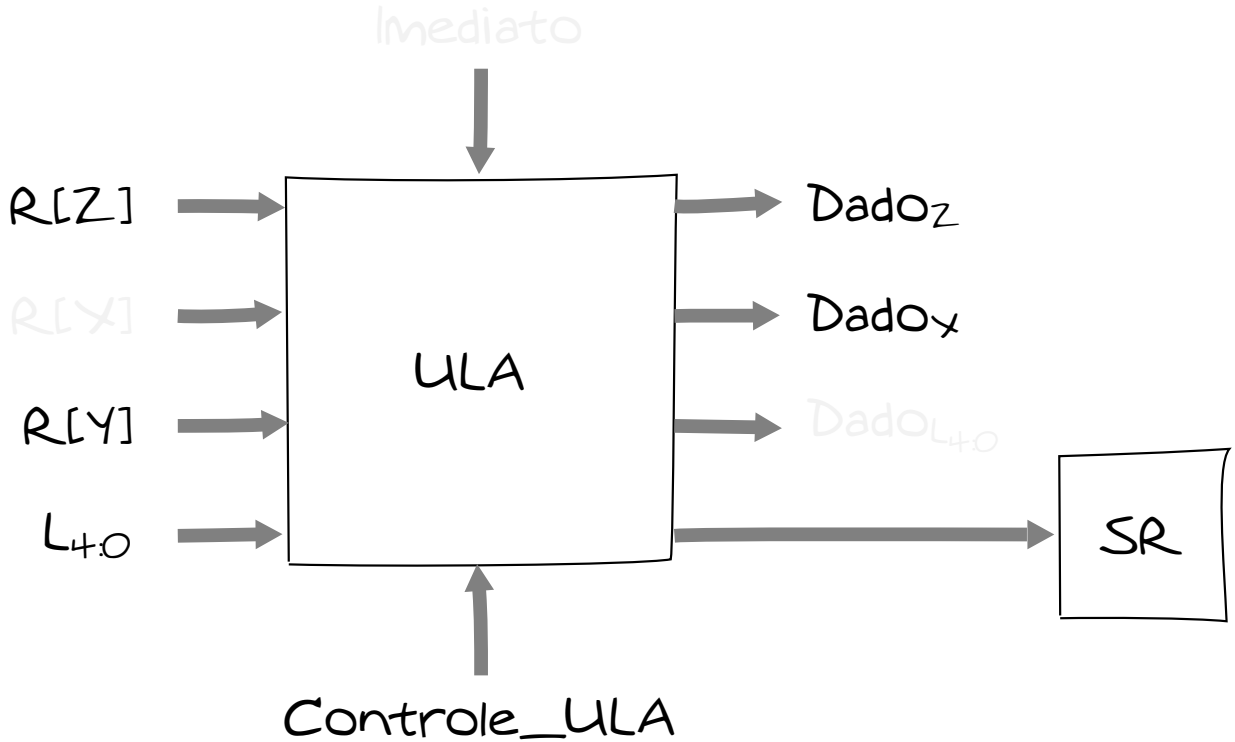
- Executar o comportamento
  - Unidade de Lógica e Aritmética (ULA)



Realiza as operações aritméticas, lógicas e bit a bit, além do cálculo de endereçamento para desvio de fluxo e acesso à memória para escrita e leitura de dados

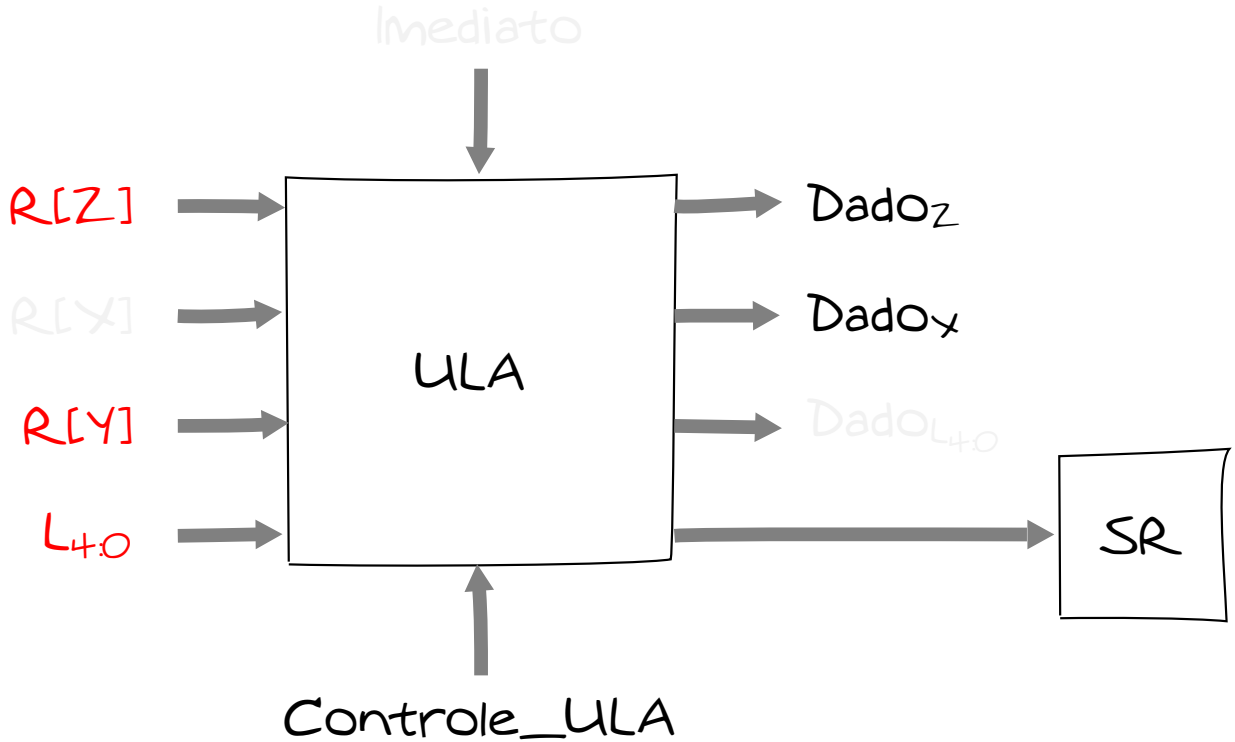
Executar operação

- Executar o comportamento
  - Deslocamento aritmética para esquerda (**sla**, tipo U)



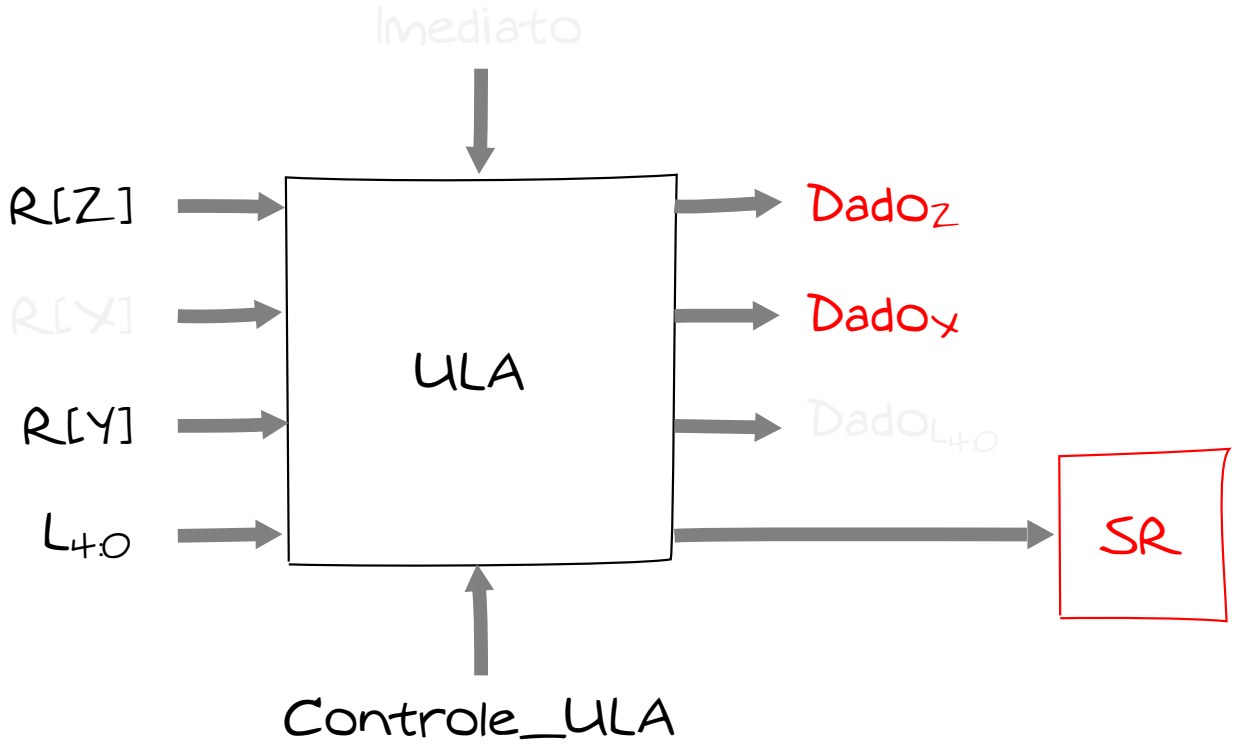
Executar operação

- Executar o comportamento
  - Deslocamento aritmética para esquerda (**sla**, tipo U)



Executar operação

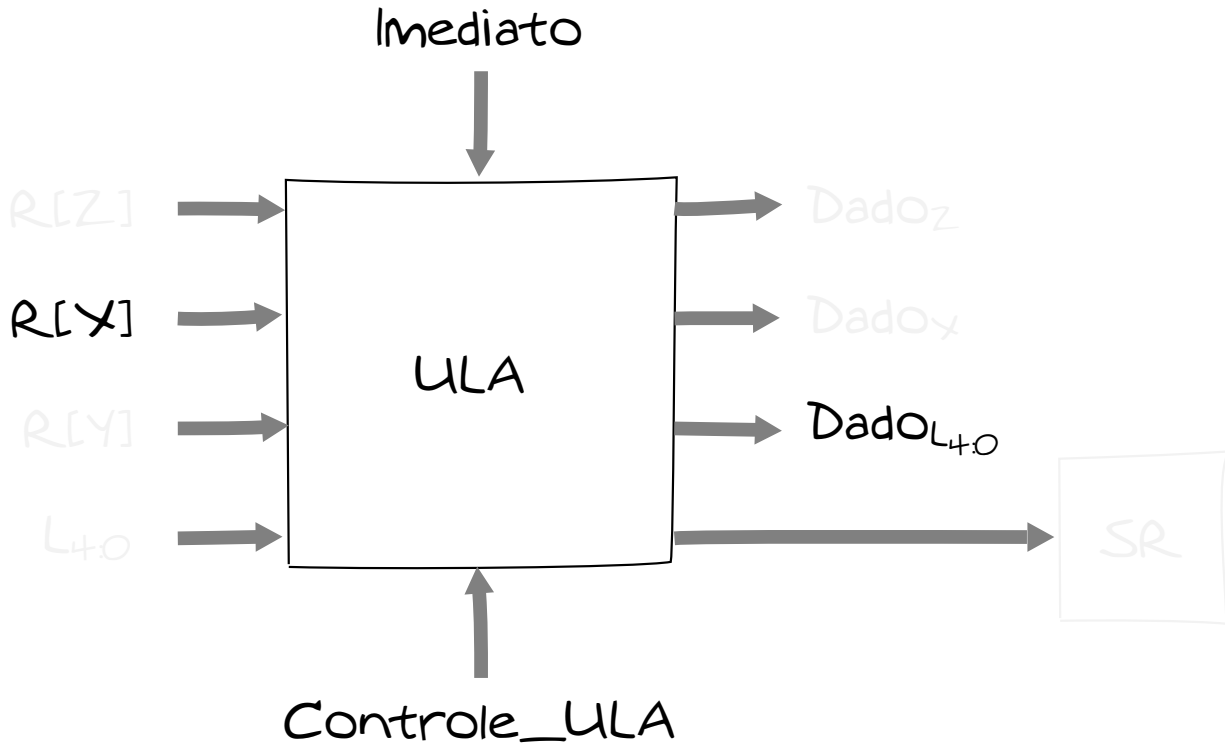
- Executar o comportamento
  - Deslocamento aritmética para esquerda (**sla**, tipo U)



O resultado da operação é atribuído em $R[z]$ e $R[x]$, utilizando os valores de $R[z]$, $R[y]$ e $L_{4:0}$ como operandos e podendo modificar alguns campos do registrador SR

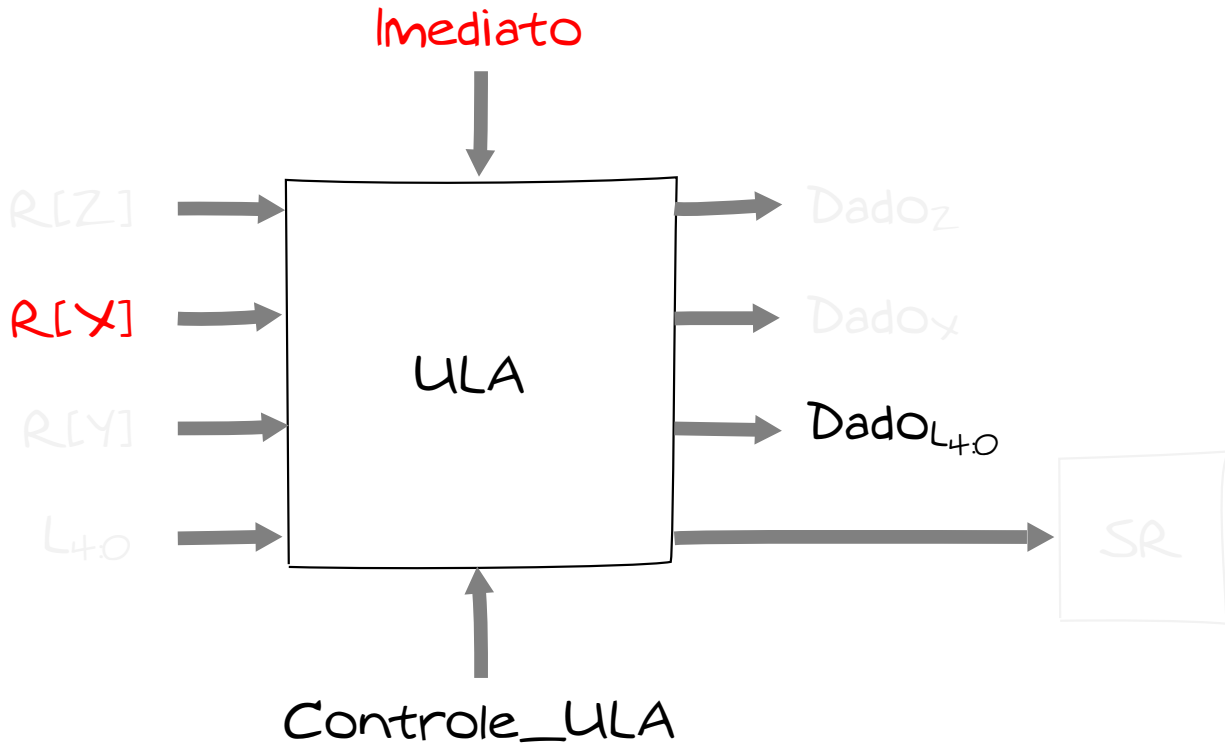
Executar operação

- Execução da instrução
  - Leitura de 32 bits da memória (**I32**, tipo F)



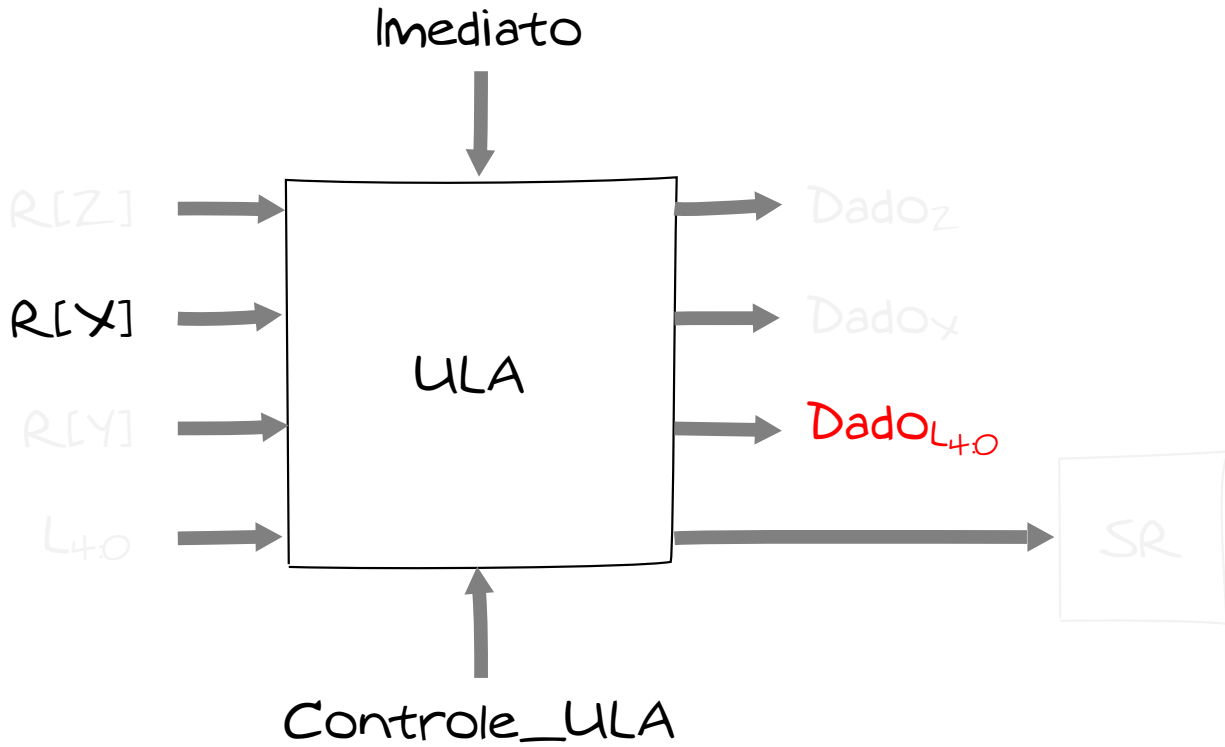
Executar operação

- Execução da instrução
  - Leitura de 32 bits da memória (**I32**, tipo F)



Executar operação

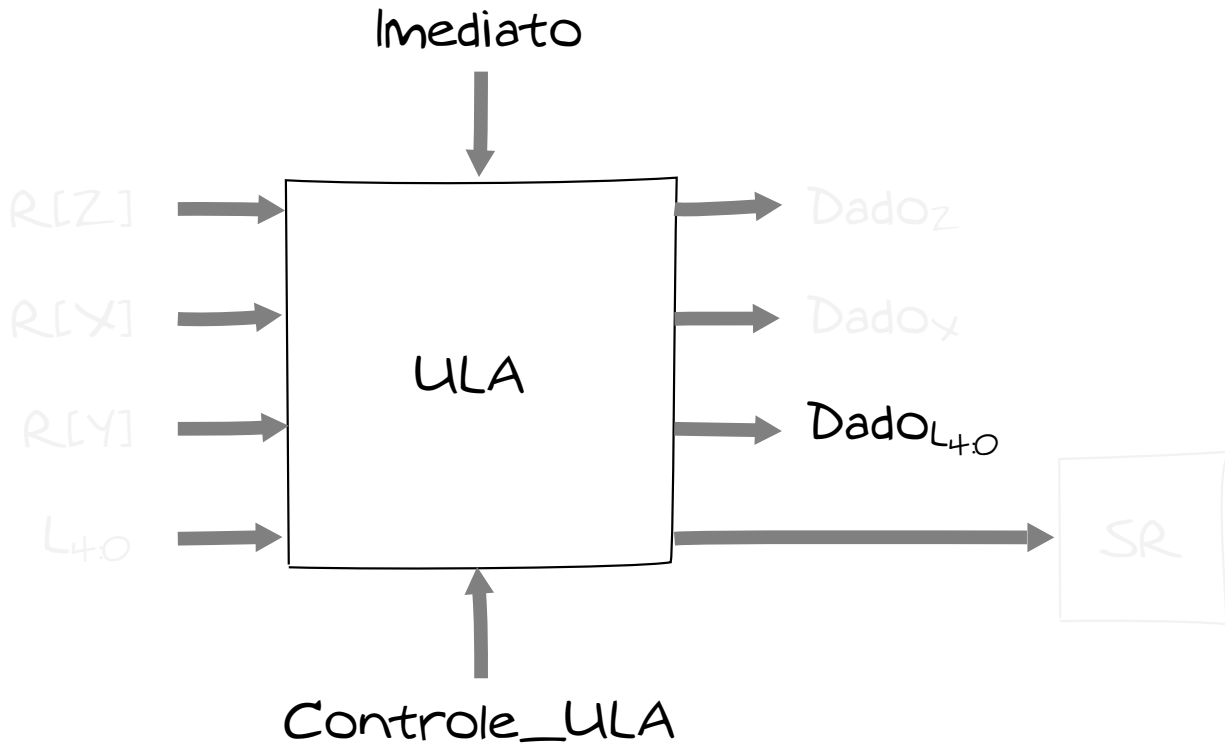
- Execução da instrução
  - Leitura de 32 bits da memória (**I32**, tipo F)



O endereço calculado é armazenado em $Dado_{L_{4:0}}$ que será utilizado no endereçamento da memória

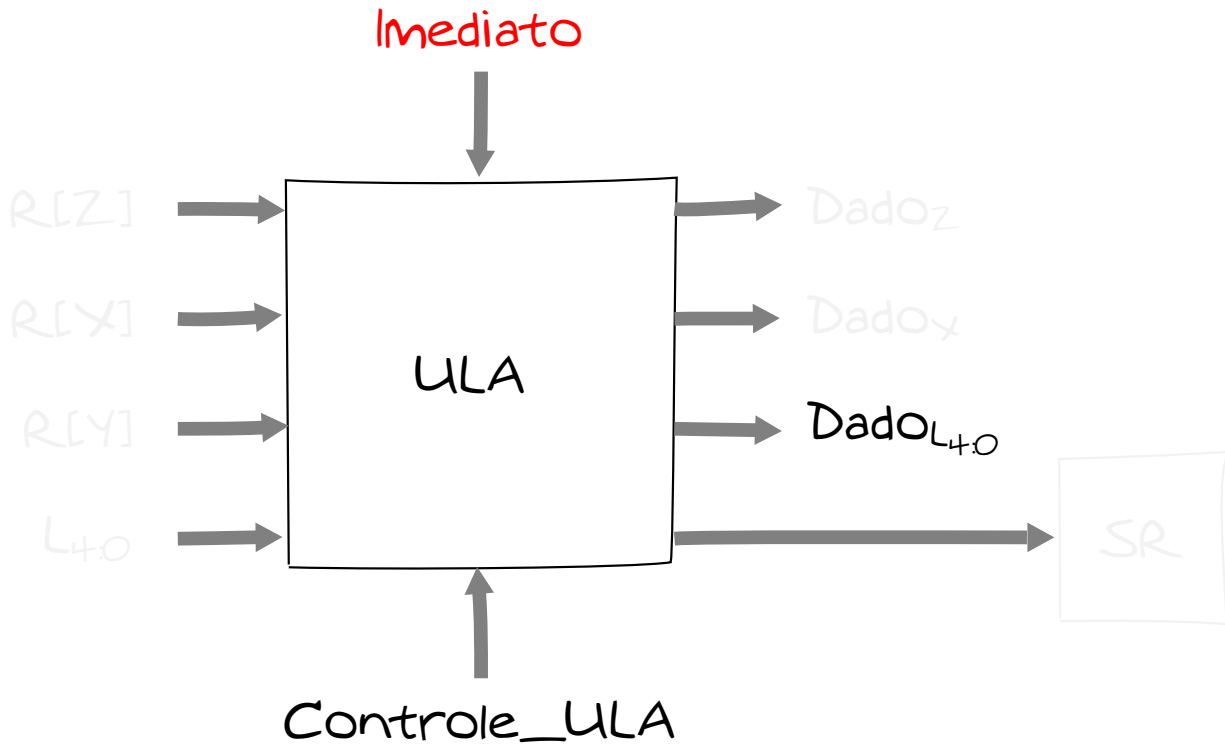
Executar operação

- Execução da instrução
  - Desvio incondicional (**bun**, tipo S)



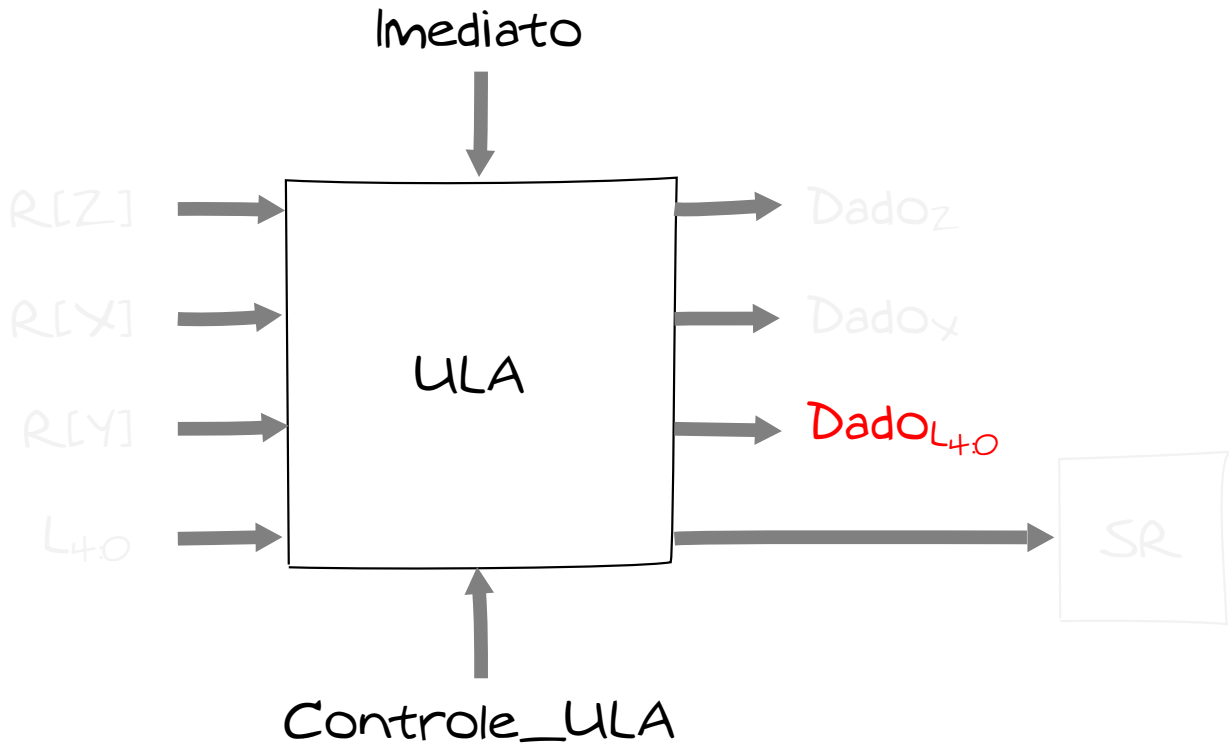
Executar operação

- Execução da instrução
  - Desvio incondicional (**bun**, tipo S)



Executar operação

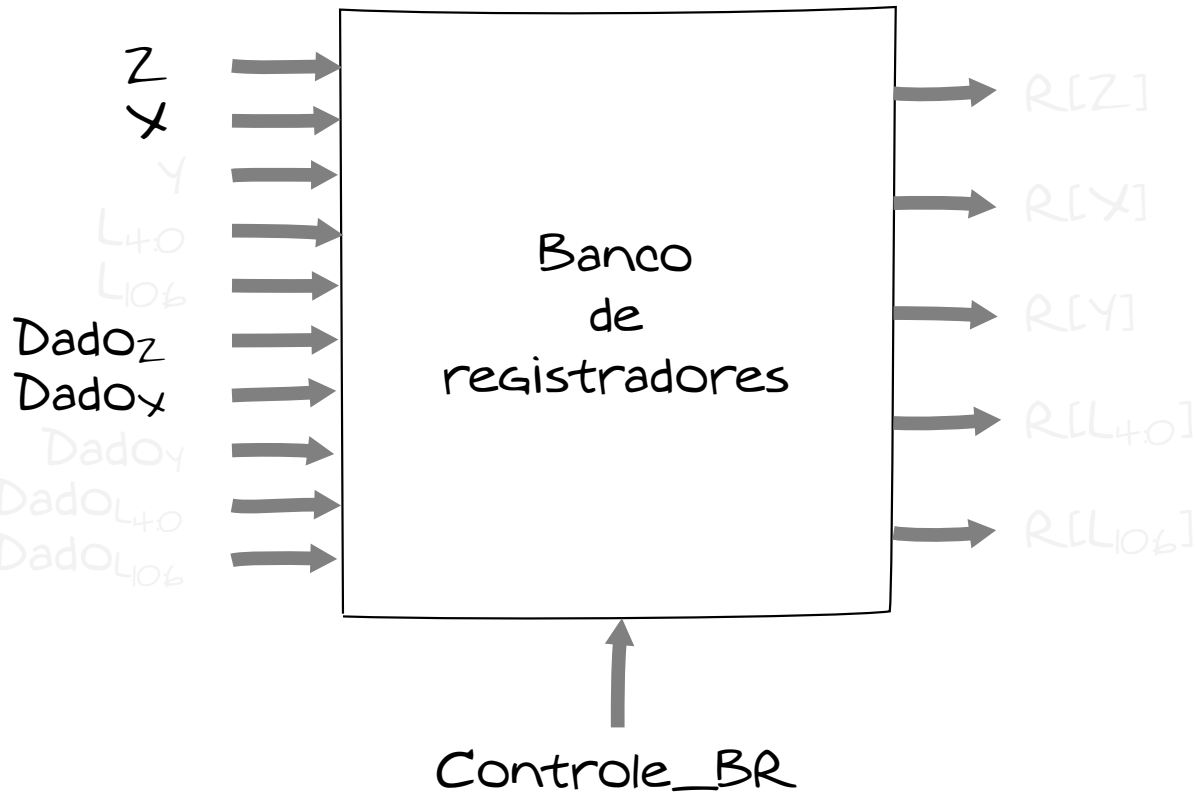
- Execução da instrução
  - Desvio incondicional (**bun**, tipo S)



O endereço do desvio é armazenado em $Dado_{L_{4:0}}$ e será atribuído ao registrador PC pelo controle

Armazenar dados

- Armazenar os resultados
  - Atribuição de valores em registradores

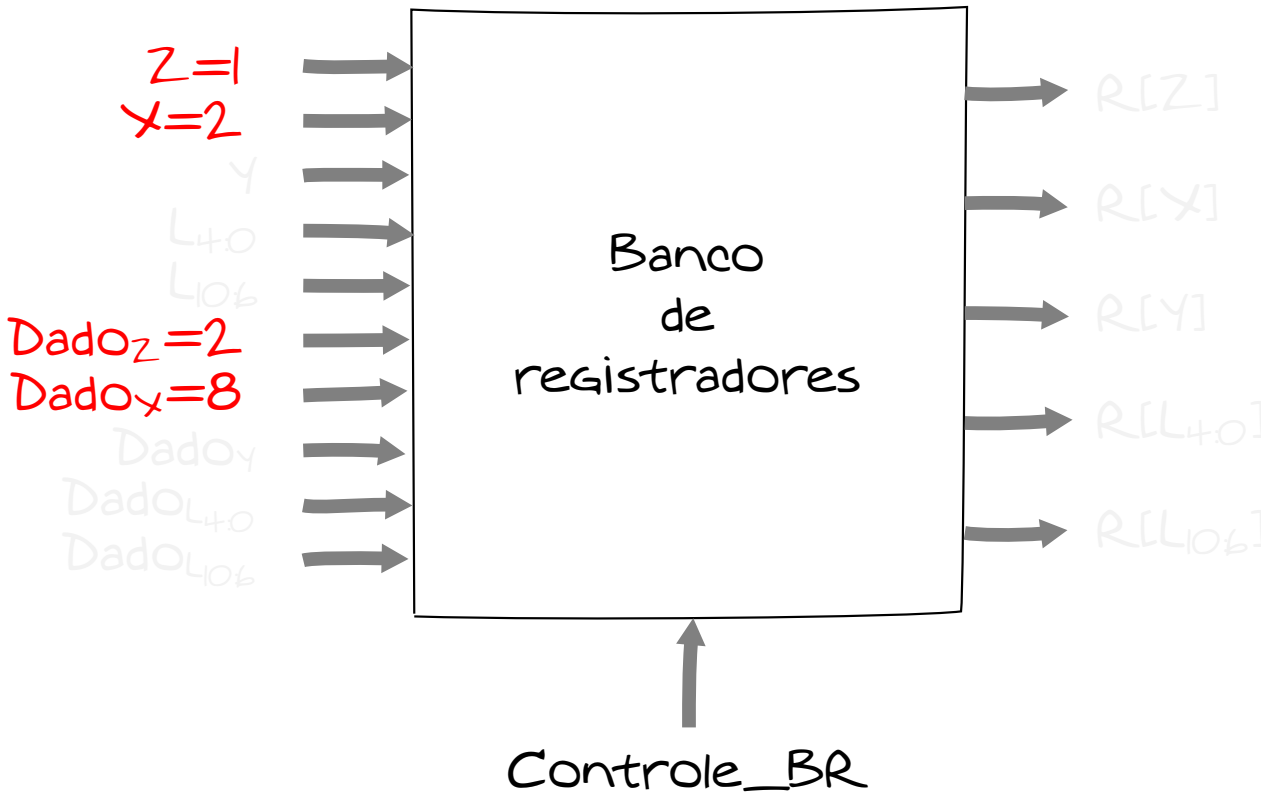


Ex: r1 = 1, r2 = 2, r3 = 4

sla r1, r2, r3, 1

Armazenar dados

- Armazenar os resultados
  - Atribuição de valores em registradores

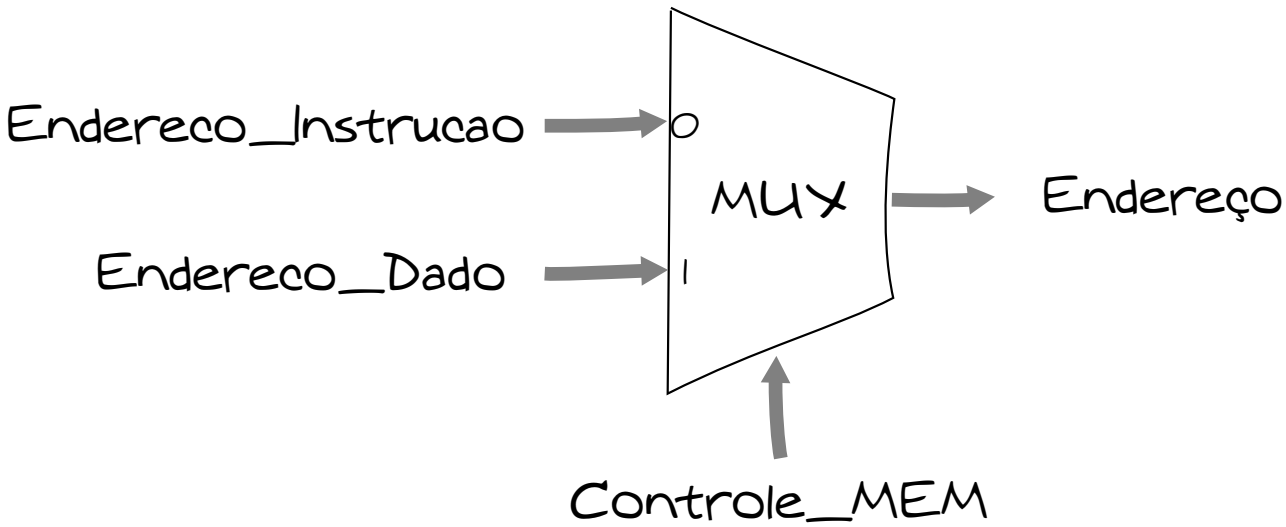


Ex: r1 = 2, r2 = 8, r3 = 4

sla r1, r2, r3, 1

Armazenar dados

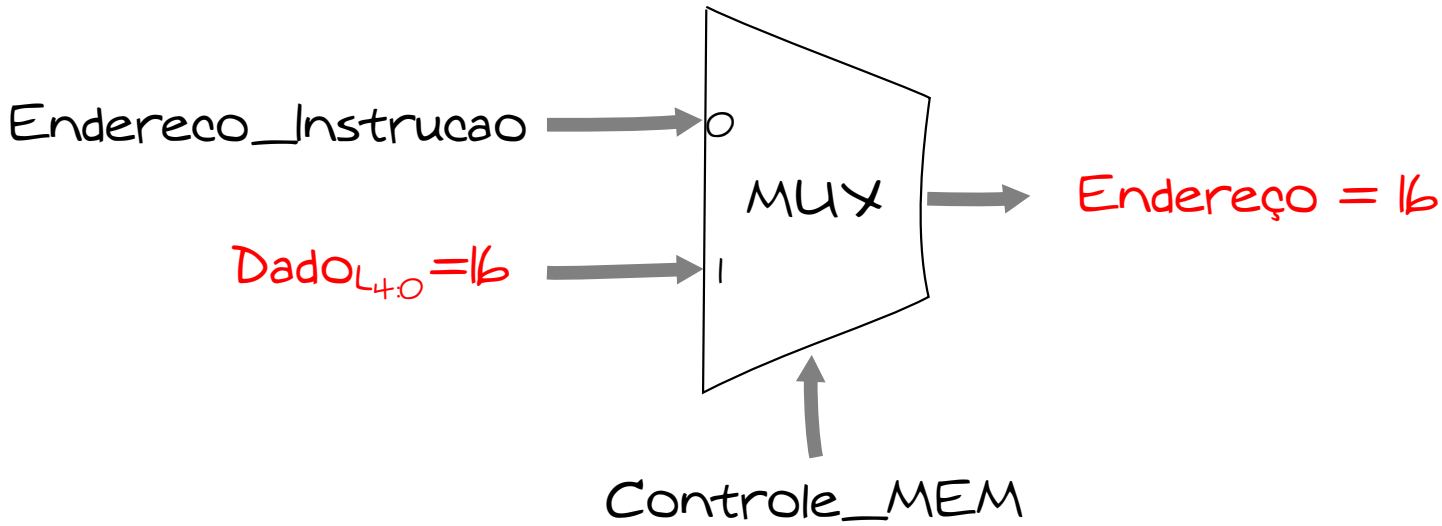
- Armazenar os resultados
  - Acesso à memória



Ex: **s32** [r0 + 4], r0

Armazenar dados

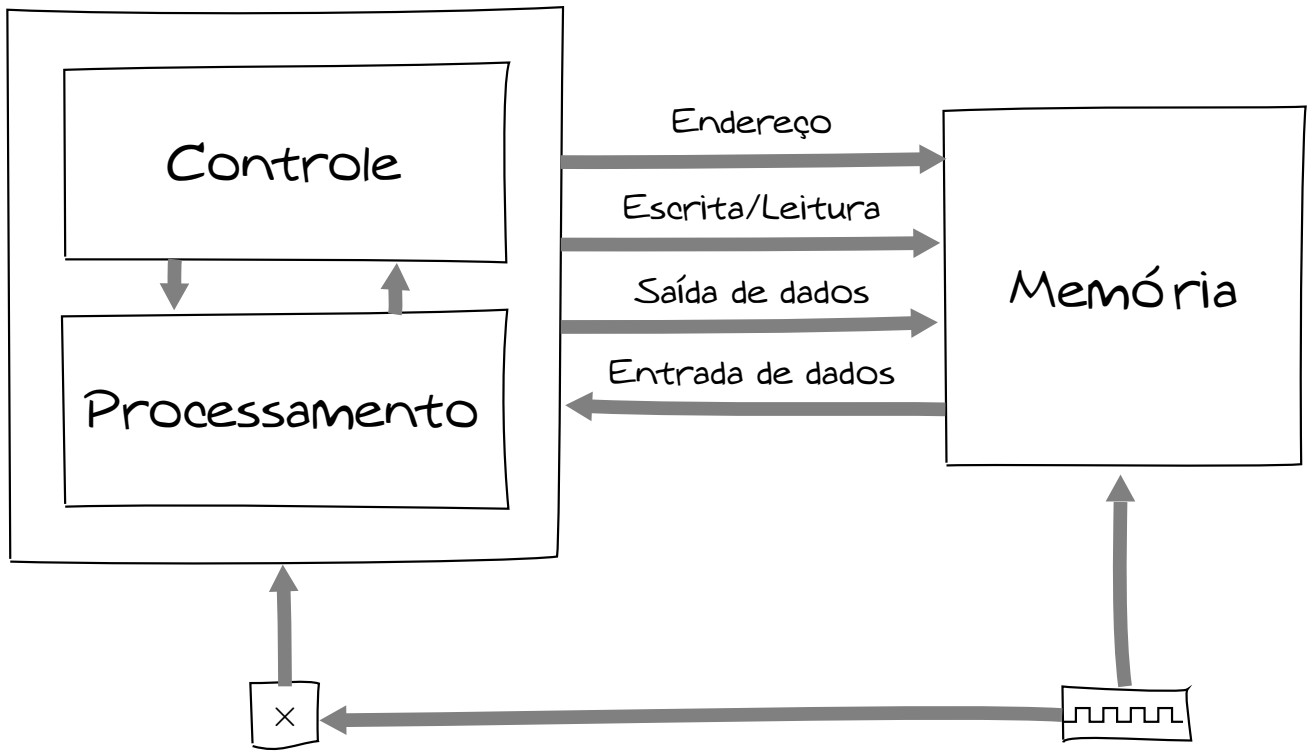
- Armazenar os resultados
  - Acesso à memória



Ex: **s32** [r0 + 4], r0

Diagrama de blocos

- Arquitetura = Controle + Processamento



Exemplo

- Considerando as instruções abaixo, execute seus respectivos comportamentos indicando a sequência de controle para ativar os caminhos de dados

```
1 // r1 = 8, r2 = 3
2 div r4, r3, r1, r2
3 sub r5, r1, r4
4 push r1, r2, r3, r4, r5
```