Organização e Arquitetura de Computadores I

Caminho de Dados

Sumário

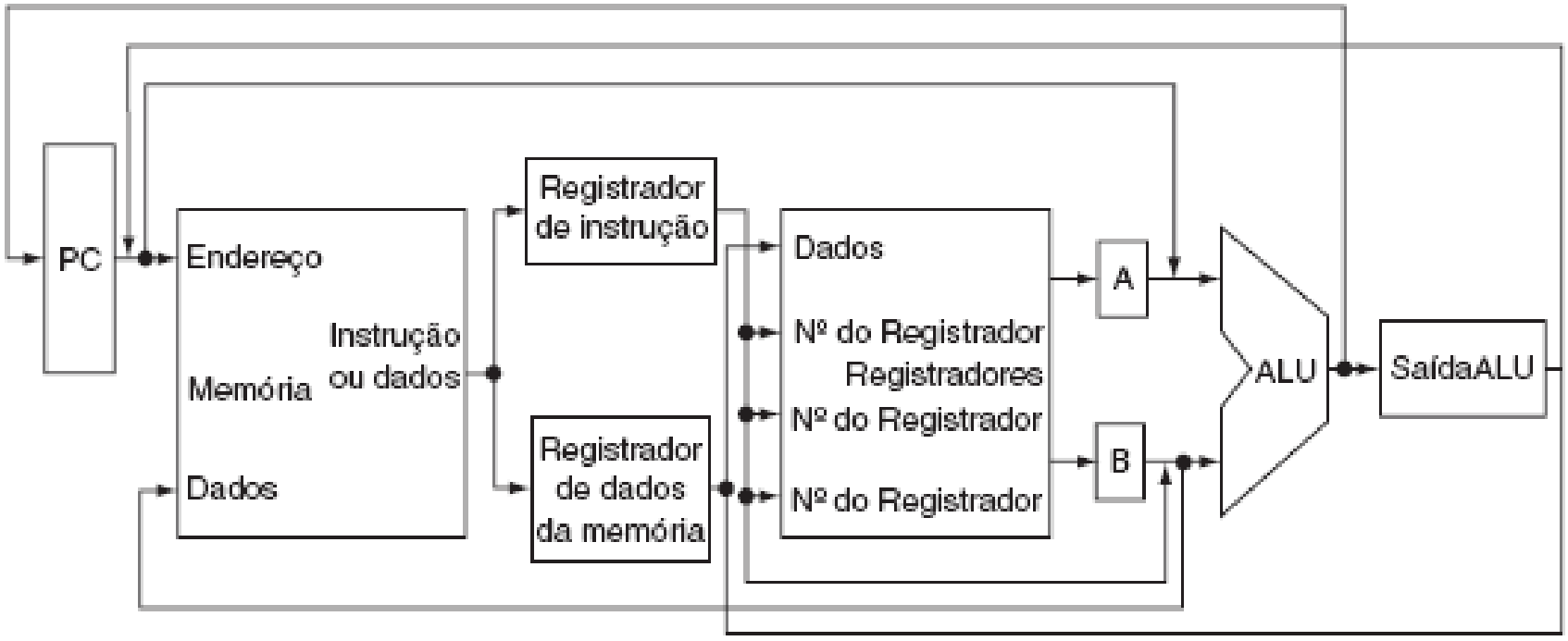
- Introdução
- Convenções Lógicas de Projeto
- Construindo um Caminho de Dados
- O Controle da ULA
- Projeto da Unidade de Controle Principal
- Operação do Caminho de Dados
  - Operação do Caminho de Dados – Tipo R
  - Operação do Caminho de Dados – Load
  - Operação do Caminho de Dados – beq
- Uma Implementação Multiciclo
- Etapas de Execução

Uma Implementação Multiciclo

- Se dividirmos cada instrução em uma série de etapas correspondentes às operações das unidades funcionais necessárias, podemos usar essas etapas para criar uma implementação multiciclo.
- Em uma implementação multiciclo, cada etapa da execução levará 1 ciclo de *clock*.
- A implementação multiciclo permite que uma unidade funcional seja usada mais de uma vez por instrução, desde que seja usada em diferentes ciclos de *clock*.

Uma Implementação Multiciclo

- Visão de alto nível de um caminho multiciclos:



Uma Implementação Multiciclo

- Uma única unidade de memória é usada para instruções e para dados.
- Existe uma única ULA, em vez de uma ULA e dois somadores.
- Um ou mais registradores são adicionados após cada unidade funcional para conter a saída dessa unidade, até o valor ser usado em um ciclo de *clock* subseqüente.
- No final de um ciclo de *clock*, todos os dados usados nos ciclos de *clock* subseqüentes precisam ser armazenados em um elemento de estado, visível ao programador: banco de registradores, o PC ou a memória.

Uma Implementação Multiciclo

- Os dados usados pela mesma instrução em um ciclo posterior precisam ser armazenados em um desses registradores adicionais.
- A posição dos registradores adicionais é determinada por dois fatores:
  - Que unidades combinacionais cabem em um ciclo de *clock*;
  - Que dados são necessários em ciclos posteriores implementando a instrução.

Uma Implementação Multiciclo

- Somente uma das seguintes operações pode ser executada em um ciclo de *clock*:
  - Um acesso à memória;
  - Um acesso ao banco de registradores (duas leituras e uma escrita);
  - Uma operação da ULA.
- Quaisquer dados produzidos por uma dessas três unidades funcionais precisam ser salvos em um registrador temporário para uso em um ciclo posterior.

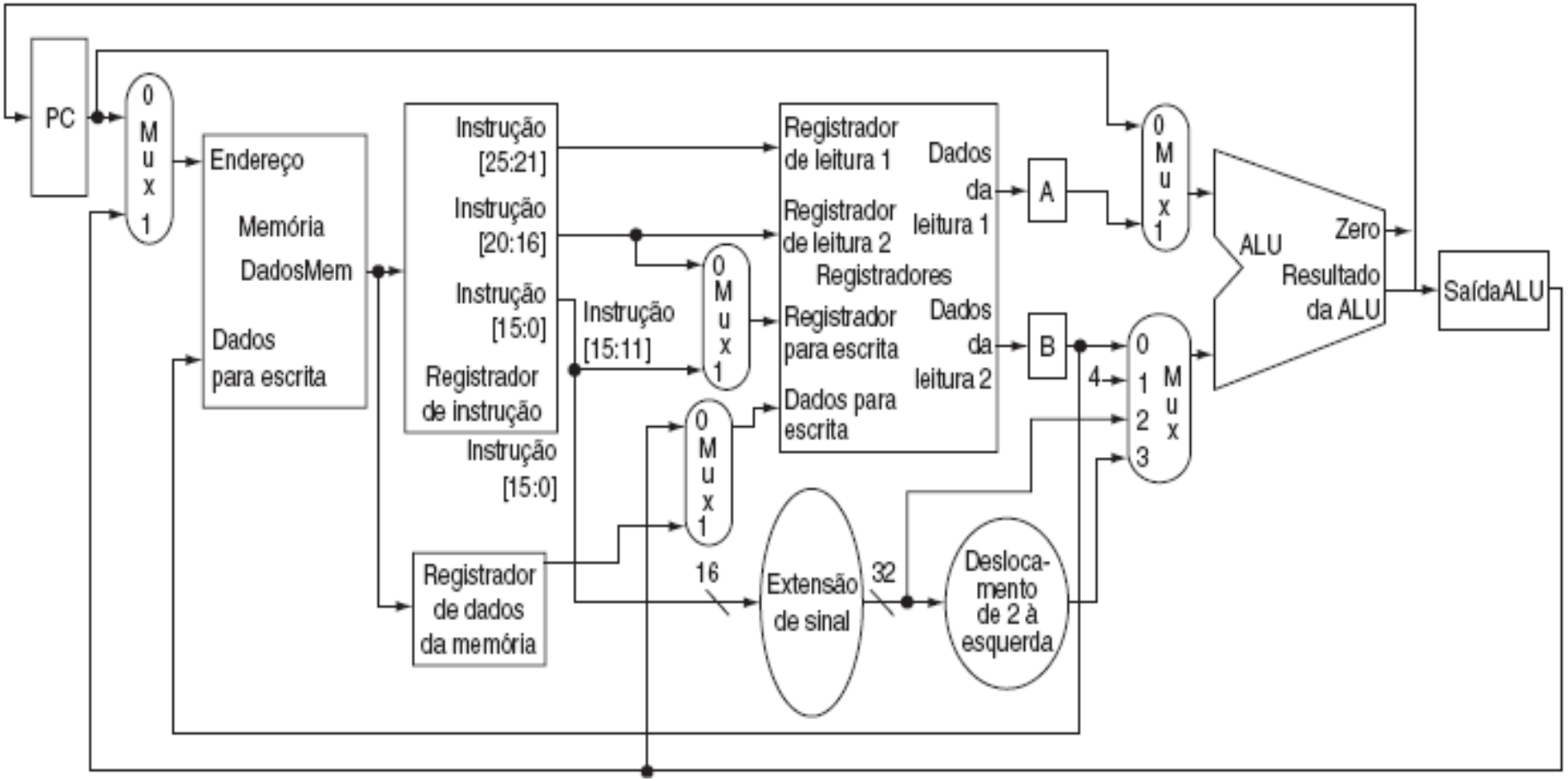
Uma Implementação Multiciclo

- Registradores temporários adicionados:
  - O **registrador de instrução** (**IR**) e o **registrador de dados da memória** (**MDR**) são incluídos para salvar a saída da memória para uma leitura de instrução e uma leitura de dados, respectivamente;
  - Os registradores **A** e **B** são usados para conter os valores dos registradores operandos lidos do banco de registradores;
  - O registrador **saídaALU** contém a saída da ULA.
- Todos os registradores exceto o IR contêm dados apenas entre um par de ciclos de *clock* adjacentes e, portanto, não precisarão de um sinal de controle de escrita.

Uma Implementação Multiciclo

- Como várias unidades funcionais são compartilhadas para diferentes finalidades, precisamos de ambos: incluir multiplexadores e expandir os multiplexadores existentes.

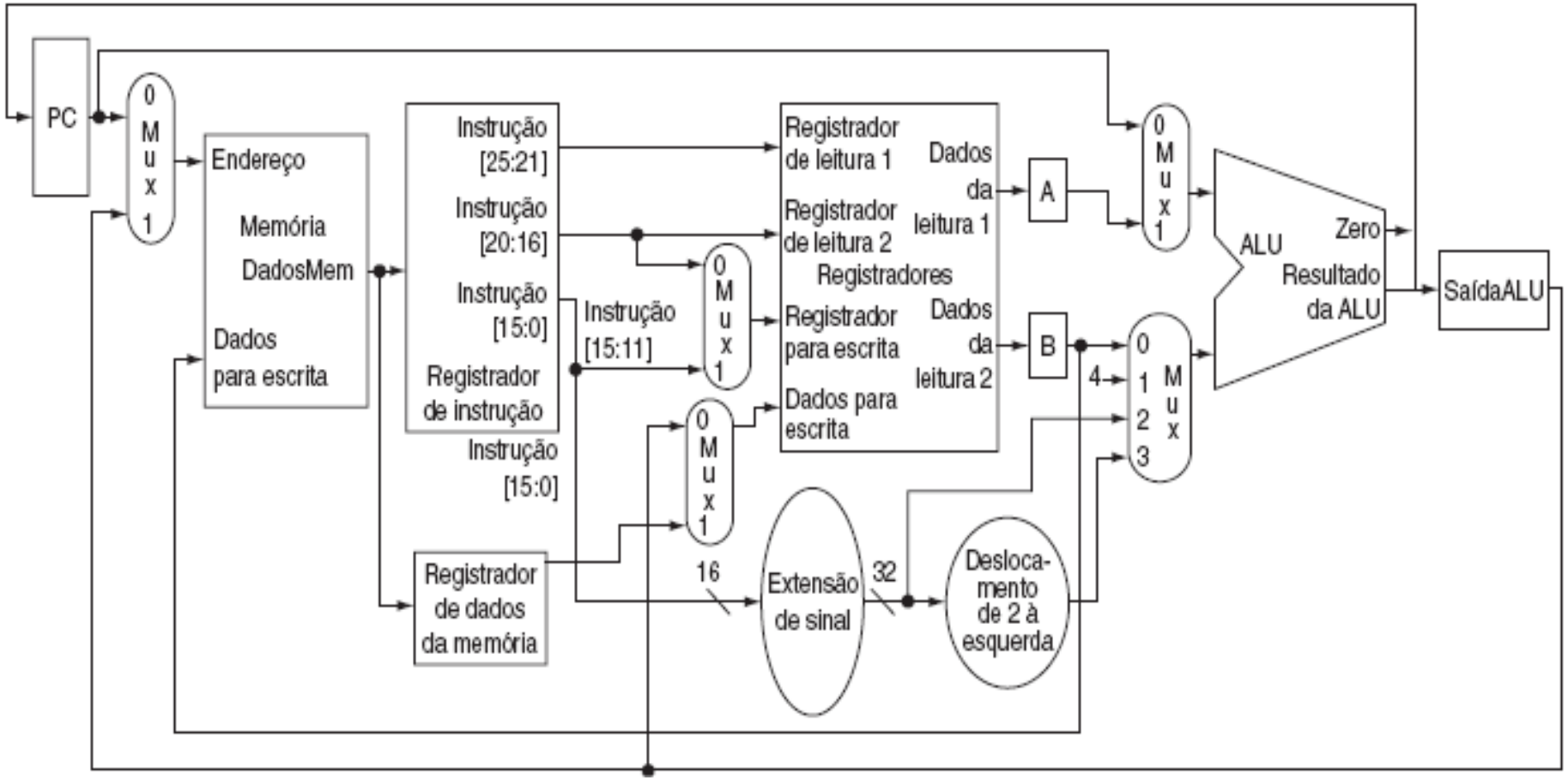
Uma Implementação Multiciclo



Uma Implementação Multiciclo

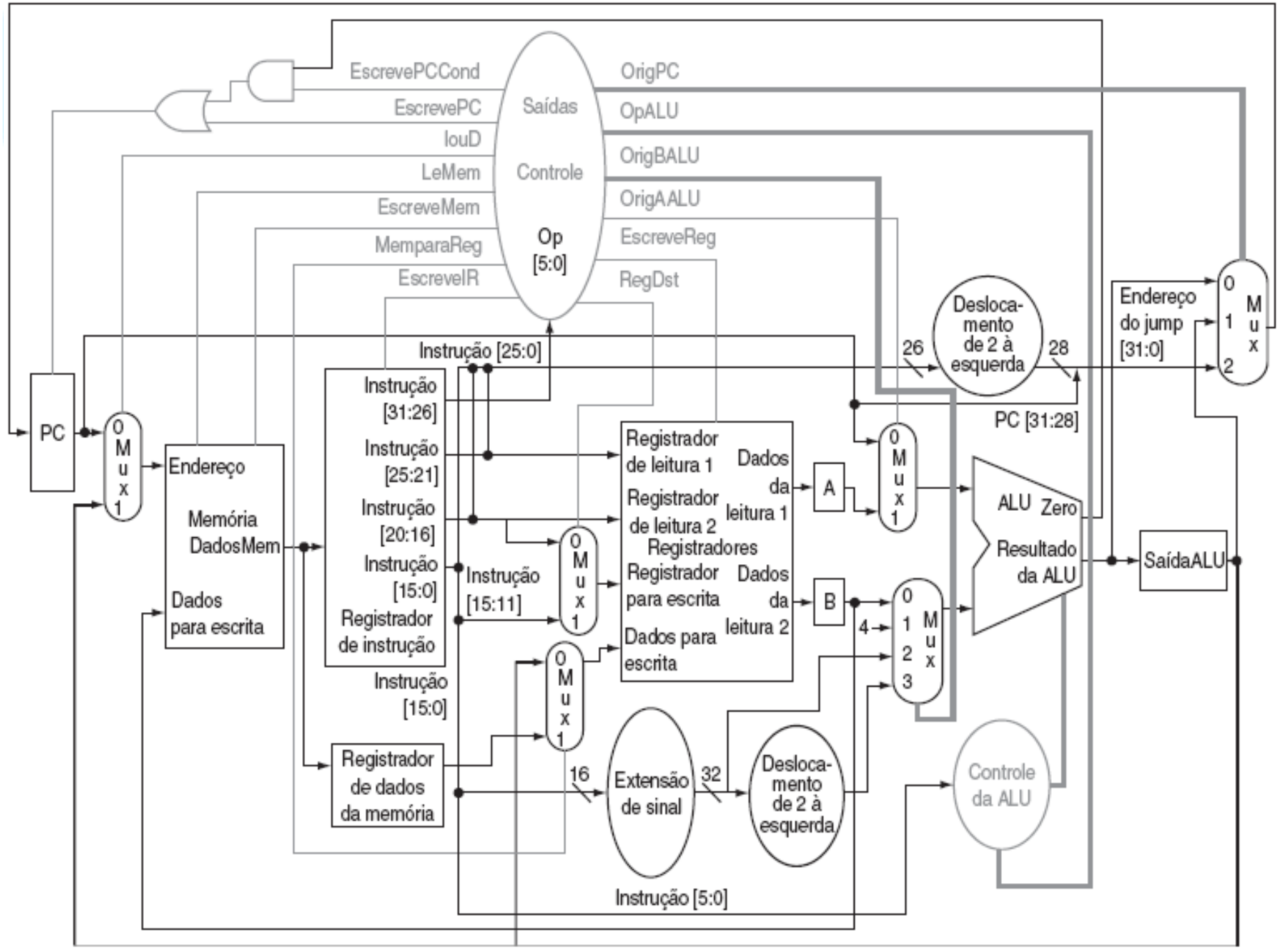
- Substituir as três ULAs do caminho de dados de ciclo único por uma única ULA significa que a única ULA precisa acomodar todas as entradas que, antes, iam para as três ULAs diferentes. Para tanto é necessário:
  - Um multiplexador adicional é incluído para a primeira entrada da ULA (registrador A e PC);
  - O multiplexador na segunda entrada da ULA muda de duas para quatro entradas. Acrescenta uma para o valor 4 e uma para o campo *offset* com sinal estendido e deslocado.

Uma Implementação Multiciclo



Uma Implementação Multiciclo

- Como o caminho de dados atual usa múltiplos ciclos de *clock* por instrução, ele exigirá um conjunto diferente de sinais de controle.
- As unidades de estado visíveis ao programador (o PC, a memória e os registradores), bem como o IR, precisarão de sinais de controle de escrita.
- A memória também precisará de um sinal de leitura.
- A unidade controle da ULA será a mesma utilizada em um caminho de ciclo único.
- Cada multiplexador de quatro entradas exige duas linhas de controle e cada multiplexador de duas entradas exige uma linha de controle.


EscrevePCCond
EscrevePC
IouD
LeMem
EscreveMem
MemparaReg
EscreveIR
Saídas
Controle
Op
[5:0]
OrigPC
OpALU
OrigBALU
OrigAALU
EscreveReg
RegDst
Instrução [25:0]
26
Deslocamento de 2 à esquerda
28
Endereço do jump [31:0]
PC [31:28]
PC
Endereço
Memória
DadosMem
Dados para escrita
Instrução [31:26]
Instrução [25:21]
Instrução [20:16]
Instrução [15:0]
Registrador de instrução
Instrução [15:0]
Instrução [15:11]
Registrador de dados da memória
Registrador de leitura 1
Registrador de leitura 2
Registradores
Registrador para escrita
Dados para escrita
Dados da leitura 1
Dados da leitura 2
A
B
4
ALU
Zero
Resultado da ALU
SaídaALU
16
Extensão de sinal
32
Deslocamento de 2 à esquerda
Controle da ALU
Instrução [5:0]
Mux

Uma Implementação Multiciclo

Nome do sinal	Valor	Efeito
OpALU	00	A ULA efetua uma adição
	01	A ULA efetua uma subtração
	10	O campo funct da instrução determina a operação
OrigBALU	00	A segunda entrada da ULA vem do registrador B
	01	A segunda entrada da ULA é a constante 4
	10	A segunda entrada da ULA são os 16 bits menos significativos com sinal estendido do IR
	11	A segunda entrada da ULA são os 16 bits menos significativos com sinal estendido do IR deslocados de 2 bits para a esquerda
OrigPC	00	A saída da ULA (PC +4) é enviada ao PC para escrita
	01	O conteúdo da SaídaALU (endereço de destino do desvio) é enviado ao PC para escrita
	10	O endereço de destino do jump (IR[25:0]) deslocado de 2 bits para a esquerda e concatenado com PC + 4[31:28] é enviado ao PC para escrita

Uma Implementação Multiciclo

- Com a instrução “j” e a instrução “beq”, há três origens possíveis para o valor a ser escrito no PC:
  - A saída da ULA, que é o valor PC + 4 durante a busca da instrução, armazenado direto no PC;
  - O registrador **saídaALU**, que é onde armazenaremos o endereço de destino do desvio após ele ser calculado;
  - Os 26 bits menos significativos do registrador de instrução (IR) deslocados de 2 à esquerda e concatenados com os 4 bits mais significativos do PC incrementado, que é a origem quando a instrução é um *jump*.
- Serão necessários dois sinais de controle separados:
  - **EscrevePC**, que causa uma escrita incondicional no PC;
  - **EscrevePCCond**, que causa uma escrita no PC se a condição de desvio também for verdadeira.

Etapas de Execução

- O objetivo em dividir a execução em ciclos de *clock* visa a maximização do desempenho.
- A instrução será dividida em uma série de etapas, cada uma usando um ciclo de *clock*, tentando manter a quantidade de trabalho por ciclo aproximadamente igual. Por exemplo:
  - Um acesso à memória;
  - Uma operação da ULA;
  - Um acesso ao banco de registradores.

Etapas de Execução

- O ciclo de *clock* pode ser tão curto quanto a mais longa operação a ser executada em uma etapa.
- Ao final de cada ciclo de *clock*, quaisquer valores de dados necessários em um ciclo subsequente precisam ser armazenados em um registrador.
- Devido ao sistema ser acionado por transição, o valor de um registrador só muda com a transição do ciclo do *clock*.
- Cada instrução usa um conjunto de elementos do caminho de dados para realizar sua execução.

Etapas de Execução

- No caminho de dados de ciclo único, muitos dos elementos do caminho de dados operam em série, usando a saída de outro elemento como entrada.Outros operam em paralelo.
- No caminho de dados multiciclo, todas as operações listadas em uma etapa ocorrem em paralelo dentro de um ciclo de *clock*. Enquanto etapas sucessivas ocorrem e diferentes ciclos de *clock*.

Etapas de Execução

- Existe distinção entre ler do ou escrever no PC ou em um dos registradores independentes e ler do ou escrever no banco de registradores.
  - No primeiro caso, a ação de ler ou escrever é parte de um ciclo de clock;
  - Enquanto ler ou escrever um resultado no banco de registradores exige um ciclo de *clock* adicional.
- O banco de registradores possui *overhead* adicional de controle e acesso se comparado com os registradores únicos independentes.

Etapas de Execução

1.Etapa de busca da instrução

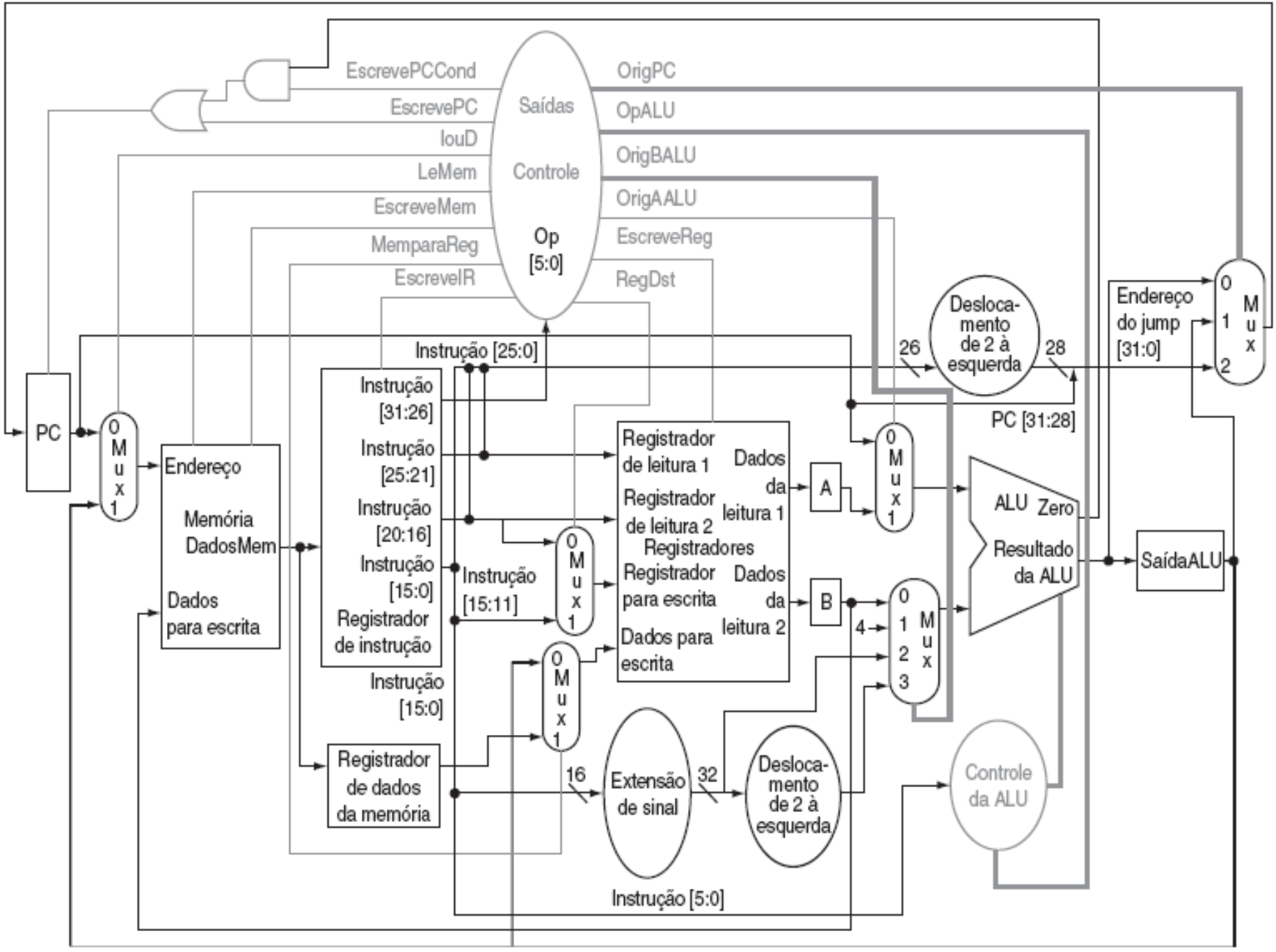
- Buscar a instrução na memória e calcular o endereço da próxima instrução sequêncial.
- Descrição na RTL (*Register-Transfer Language*)

IR <= Memória[PC];

PC <= PC + 4;

- Operação:
  - Enviar o PC para a memória como o endereço, realizar uma leitura e escrever a instrução no Registro de Instrução (IR), onde ele será armazenado.

RegDst	Escreve Reg	OrigAALU	LeMem	Escreve Mem	MemparaReg	IouD	EscrevelR	Escreve PC	Escreve PCCond	OpALU	OrigBALU	OrigPC
		0	1			0	1	1		00	01	00


EscrevePCCond
EscrevePC
IouD
LeMem
EscreveMem
MemparaReg
EscreveIR
Saídas
Controle
Op
[5:0]
OrigPC
OpALU
OrigBALU
OrigAALU
EscreveReg
RegDst
Instrução [25:0]
26
Deslocamento de 2 à esquerda
28
PC [31:28]
Endereço do jump [31:0]
PC
Endereço
Memória
DadosMem
Dados para escrita
Instrução [31:26]
Instrução [25:21]
Instrução [20:16]
Instrução [15:0]
Registrador de instrução
Instrução [15:11]
Instrução [15:0]
Registrador de dados da memória
Registrador de leitura 1
Registrador de leitura 2
Registradores
Registrador para escrita
Dados para escrita
Dados da leitura 1
Dados da leitura 2
A
B
4
ALU
Zero
Resultado da ALU
SaídaALU
16
Extensão de sinal
32
Deslocamento de 2 à esquerda
Controle da ALU
Instrução [5:0]

Etapas de Execução

1.Etapa de busca da instrução

- O incremento do PC e o acesso à memória de instruções podem ocorrer em paralelo.
- O novo valor do PC não é visível até o próximo ciclo de *clock*.

RegDst	Escreve Reg	OrigAALU	LeMem	Escreve Mem	MemparaReg	IouD	EscrevelR	Escreve PC	Escreve PCCond	OpALU	OrigBALU	OrigPC
		0	1			0	1	1		00	01	00

Etapas de Execução

2.Etapa de codificação da instrução e busca dos registradores

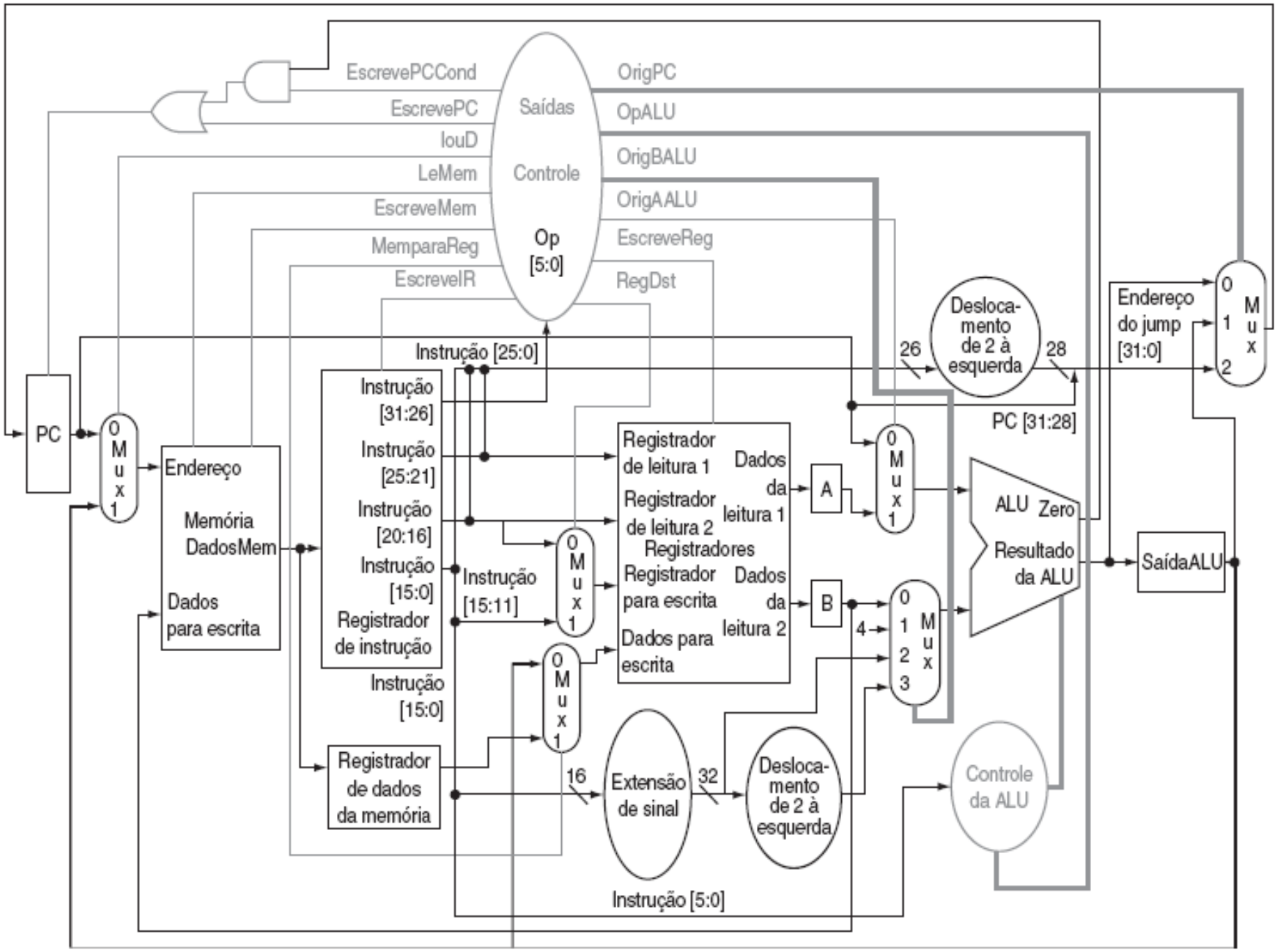
- Nesta etapa ainda não foi identificada qual é a instrução e, portanto, só são realizadas ações aplicáveis a todas as instruções ou que não sejam nocivas, caso a instrução não seja o que se espera:
  - Acessar o banco de registradores, ler os registradores rs e rt, e os armazenar nos registradores A e B;
  - Calcular o endereço de destino do desvio com a ULA, o resultado é salvo na SaídaALU (útil se a instrução for um desvio condicional);

A <= Reg[IR[25:21]];

B <= Reg[IR[20:16]];

SaídaALU <= PC + (estende-sinal (IR[15-0] << 2);

RegDst	Escreve Reg	OrigAALU	LeMem	Escreve Mem	MemparaReg	IouD	EscrevelR	Escreve PC	Escreve PCCond	OpALU	OrigBALU	OrigPC
		0								00	11	


EscrevePCCond
EscrevePC
IouD
LeMem
EscreveMem
MemparaReg
EscreveIR
Saídas
Controle
Op
[5:0]
OrigPC
OpALU
OrigBALU
OrigAALU
EscreveReg
RegDst
Instrução [25:0]
26
Deslocamento de 2 à esquerda
28
PC [31:28]
Endereço do jump [31:0]
PC
Endereço
Memória
DadosMem
Dados para escrita
Instrução [31:26]
Instrução [25:21]
Instrução [20:16]
Instrução [15:0]
Registrador de instrução
Instrução [15:11]
Instrução [15:0]
Registrador de leitura 1
Registrador de leitura 2
Registradores
Registrador para escrita
Dados para escrita
Dados da leitura 1
Dados da leitura 2
A
B
4
ALU
Zero
Resultado da ALU
SaídaALU
Registrador de dados da memória
16
Extensão de sinal
32
Deslocamento de 2 à esquerda
Controle da ALU
Instrução [5:0]
M u x

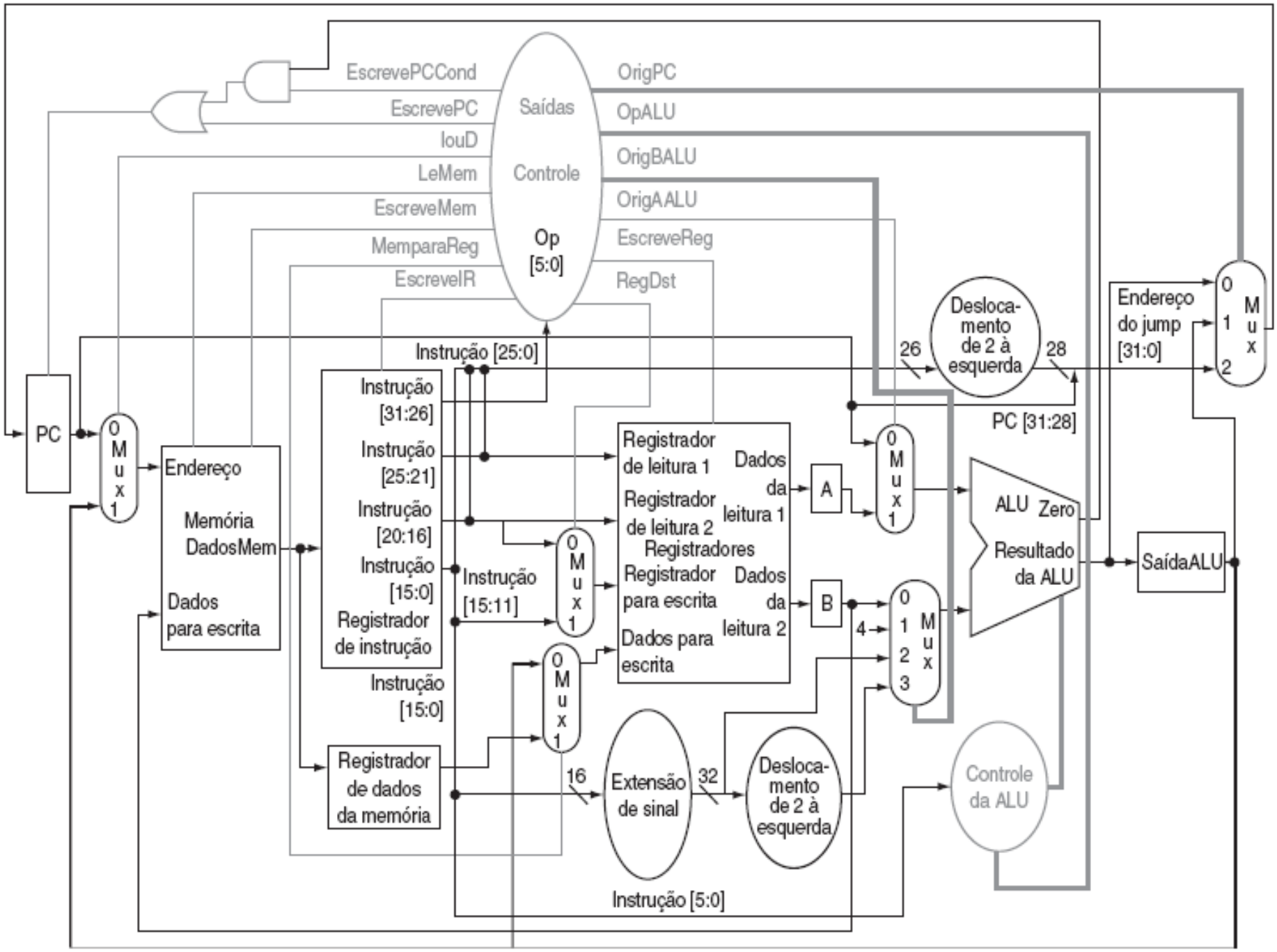
Etapas de Execução

3.Execução

- Este é o primeiro ciclo durante o qual a operação do caminho de dados é determinada pela classe de instrução.
- Referência à memória:
  - A ULA soma os operandos para formar o endereço de memória.

SaídaALU <= A + estende-sinal (IR[15:0]);

RegDst	Escreve Reg	OrigAALU	LeMem	Escreve Mem	MemparaReg	IouD	EscrevelR	Escreve PC	Escreve PCCond	OpALU	OrigBALU	OrigPC
		1								00	10	


EscrevePCCond
EscrevePC
IouD
LeMem
EscreveMem
MemparaReg
EscreveIR
Saídas
Controle
Op
[5:0]
OrigPC
OpALU
OrigBALU
OrigAALU
EscreveReg
RegDst
Instrução [25:0]
26
Deslocamento de 2 à esquerda
28
PC [31:28]
Endereço do jump [31:0]
PC
Endereço
Memória
DadosMem
Dados para escrita
Instrução [31:26]
Instrução [25:21]
Instrução [20:16]
Instrução [15:0]
Registrador de instrução
Instrução [15:11]
Instrução [15:0]
Registrador de leitura 1
Registrador de leitura 2
Registradores
Registrador para escrita
Dados para escrita
Dados da leitura 1
Dados da leitura 2
A
B
4
ALU
Zero
Resultado da ALU
SaídaALU
Registrador de dados da memória
16
Extensão de sinal
32
Deslocamento de 2 à esquerda
Controle da ALU
Instrução [5:0]
M u x

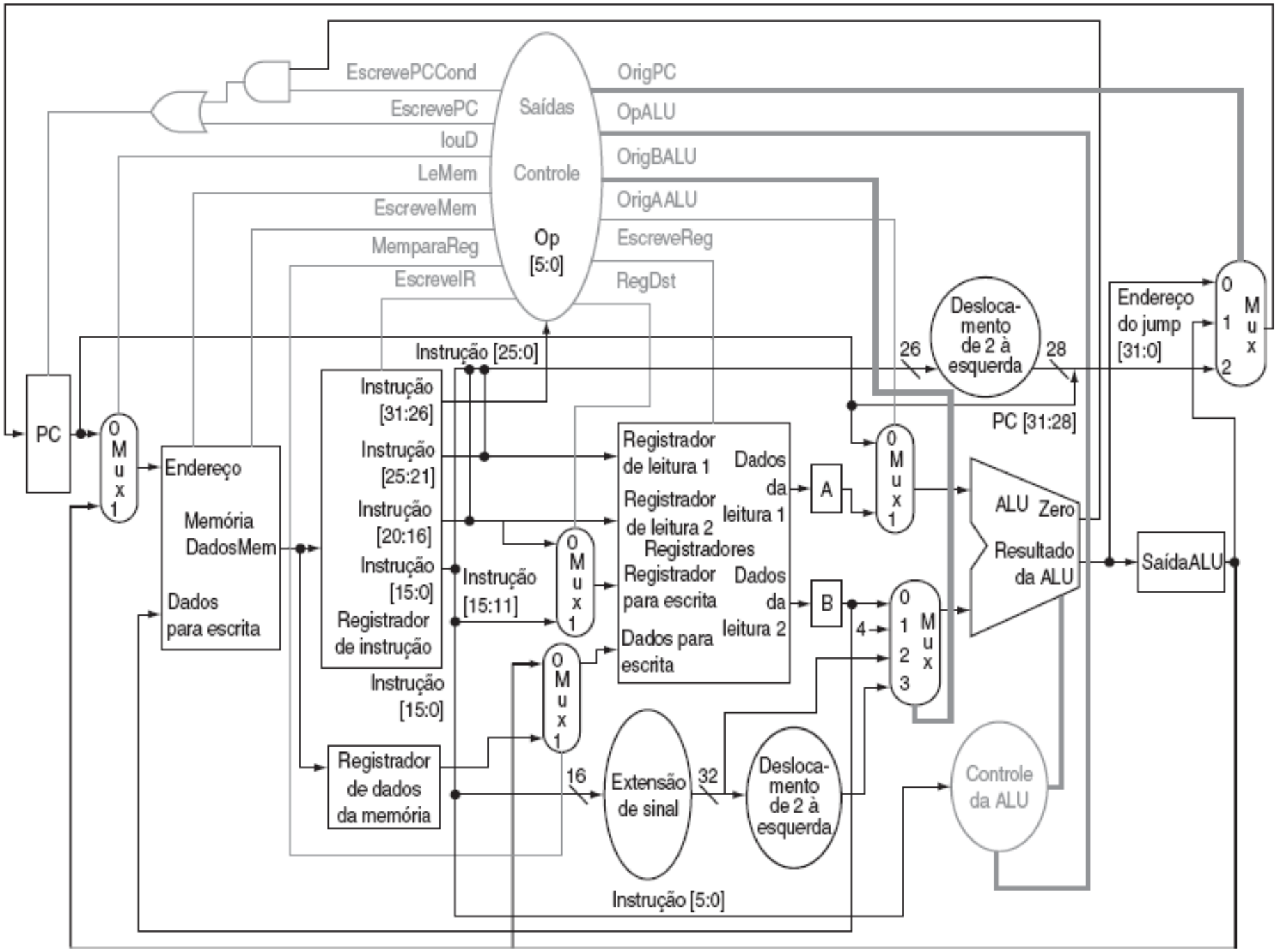
Etapas de Execução

3.Execução

- Instrução do tipo R:
  - A ULA realizará a operação especificada pelo código de função com dois valores lidos do banco de registradores do ciclo anterior.

SaídaALU <= A op B;

RegDst	Escreve Reg	OrigAALU	LeMem	Escreve Mem	MemparaReg	IouD	EscrevelR	Escreve PC	Escreve PCCond	OpALU	OrigBALU	OrigPC
		1								10	00	


EscrevePCCond
EscrevePC
IouD
LeMem
EscreveMem
MemparaReg
EscreveIR
Saídas
Controle
Op
[5:0]
OrigPC
OpALU
OrigBALU
OrigAALU
EscreveReg
RegDst
Instrução [25:0]
26
Deslocamento de 2 à esquerda
28
PC [31:28]
Endereço do jump [31:0]
PC
Endereço
Memória
DadosMem
Dados para escrita
Instrução [31:26]
Instrução [25:21]
Instrução [20:16]
Instrução [15:0]
Registrador de instrução
Instrução [15:11]
Instrução [15:0]
Registrador de leitura 1
Registrador de leitura 2
Registradores
Registrador para escrita
Dados para escrita
Dados da leitura 1
Dados da leitura 2
A
B
4
ALU
Zero
Resultado da ALU
SaídaALU
Registrador de dados da memória
16
Extensão de sinal
32
Deslocamento de 2 à esquerda
Controle da ALU
Instrução [5:0]
M u x

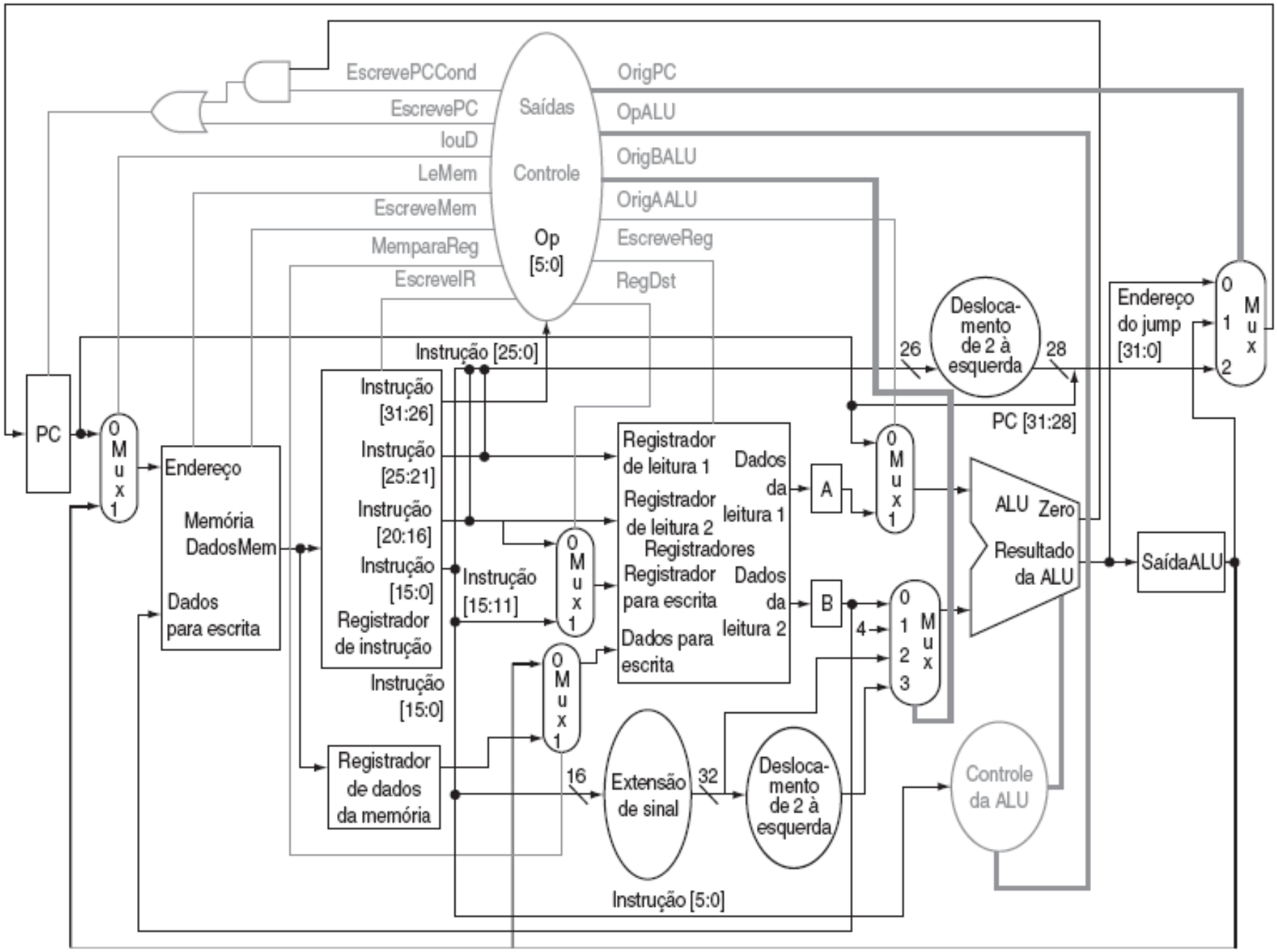
Etapas de Execução

3.Execução

- Instrução do tipo desvio (*branch*):
  - A ULA é usada para fazer a comparação de igualdade entre os dois registradores lidos na etapa anterior. O sinal zero da ULA é usado para determinar se o desvio será tomado ou não.
  - Para desvios condicionais o PC é escrito duas vezes, uma para a saída da ULA (durante a decodificação da instrução e a busca de registradores) e uma para SaídaALU (durante a etapa de conclusão do desvio).

if (A == B) PC <= SaídaALU;

RegDst	Escreve Reg	OrigAALU	LeMem	Escreve Mem	MemparaReg	IouD	EscrevelR	Escreve PC	Escreve PCCond	OpALU	OrigBALU	OrigPC
		1							1	01	00	01


EscrevePCCond
EscrevePC
IouD
LeMem
EscreveMem
MemparaReg
EscreveIR
Saídas
Controle
Op
[5:0]
OrigPC
OpALU
OrigBALU
OrigAALU
EscreveReg
RegDst
Instrução [25:0]
26
Deslocamento de 2 à esquerda
28
PC [31:28]
Endereço do jump [31:0]
PC
Endereço
Memória
DadosMem
Dados para escrita
Instrução [31:26]
Instrução [25:21]
Instrução [20:16]
Instrução [15:0]
Registrador de instrução
Instrução [15:11]
Instrução [15:0]
Registrador de leitura 1
Registrador de leitura 2
Registradores
Registrador para escrita
Dados para escrita
Dados da leitura 1
Dados da leitura 2
A
B
4
ALU
Zero
Resultado da ALU
SaídaALU
Registrador de dados da memória
16
Extensão de sinal
32
Deslocamento de 2 à esquerda
Controle da ALU
Instrução [5:0]
M u x

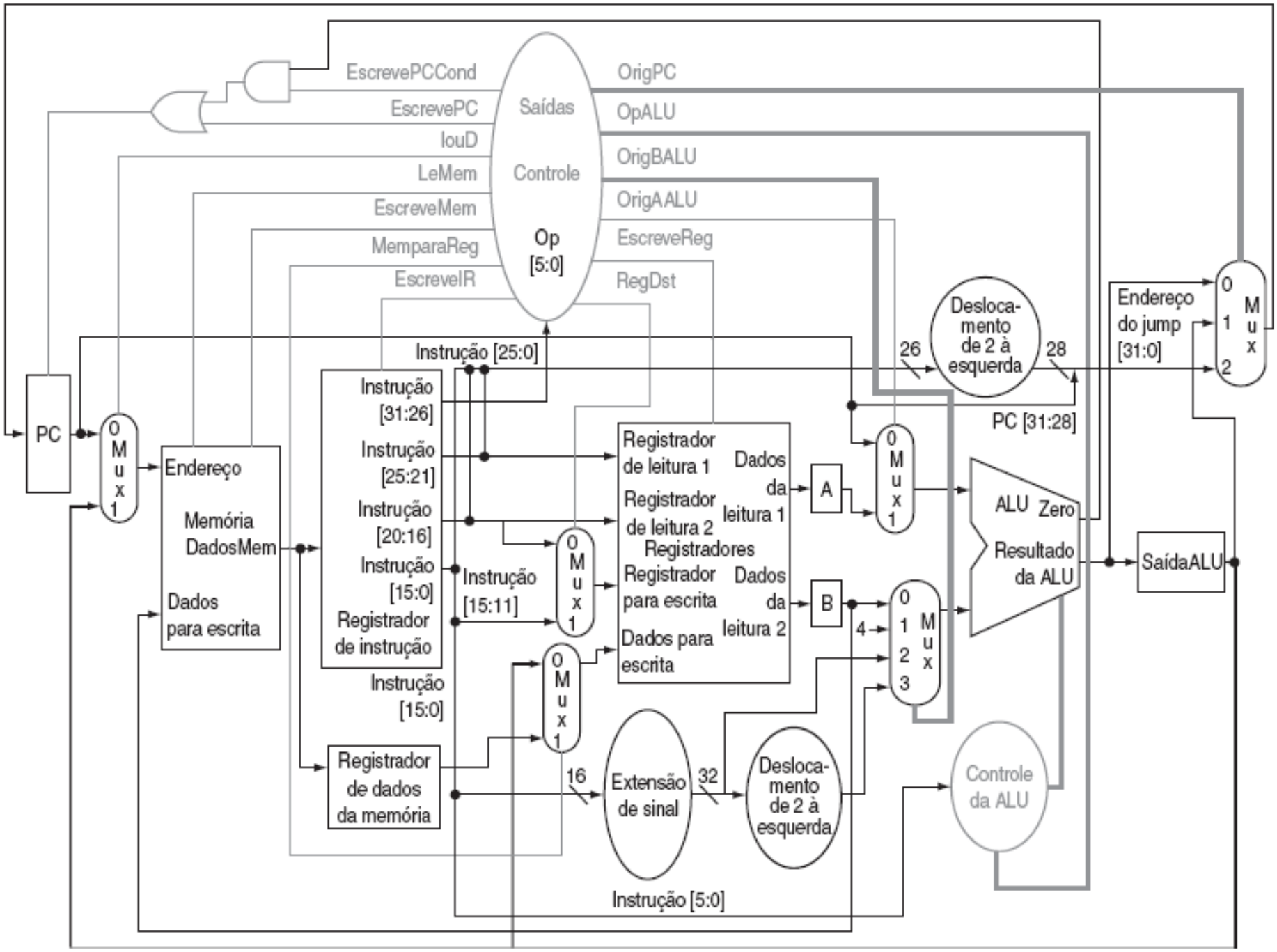
Etapas de Execução

3.Execução

- Instrução do tipo salto (*jump*):
  - O PC é substituído pelo endereço do jump. O endereço contido na instrução (26 bits), é deslocado de 2 bits à esquerda e é inserido nos 28 bits menos significativos do endereço atual e escrito no PC.

PC <= {PC [31:28], (IR [25:0], 2'b00) };

RegDst	Escreve Reg	OrigAALU	LeMem	Escreve Mem	MemparaReg	IouD	EscrevelR	Escreve PC	Escreve PCCond	OpALU	OrigBALU	OrigPC
								1				10


EscrevePCCond
EscrevePC
IouD
LeMem
EscreveMem
MemparaReg
EscreveIR
Saídas
Controle
Op
[5:0]
OrigPC
OpALU
OrigBALU
OrigAALU
EscreveReg
RegDst
Instrução [25:0]
26
Deslocamento de 2 à esquerda
28
PC [31:28]
Endereço do jump [31:0]
PC
Endereço
Memória
DadosMem
Dados para escrita
Instrução [31:26]
Instrução [25:21]
Instrução [20:16]
Instrução [15:0]
Registrador de instrução
Instrução [15:11]
Instrução [15:0]
Registrador de leitura 1
Registrador de leitura 2
Registradores
Registrador para escrita
Dados para escrita
Dados da leitura 1
Dados da leitura 2
A
B
4
ALU
Zero
Resultado da ALU
SaídaALU
Registrador de dados da memória
16
Extensão de sinal
32
Deslocamento de 2 à esquerda
Controle da ALU
Instrução [5:0]
M u x

Etapas de Execução

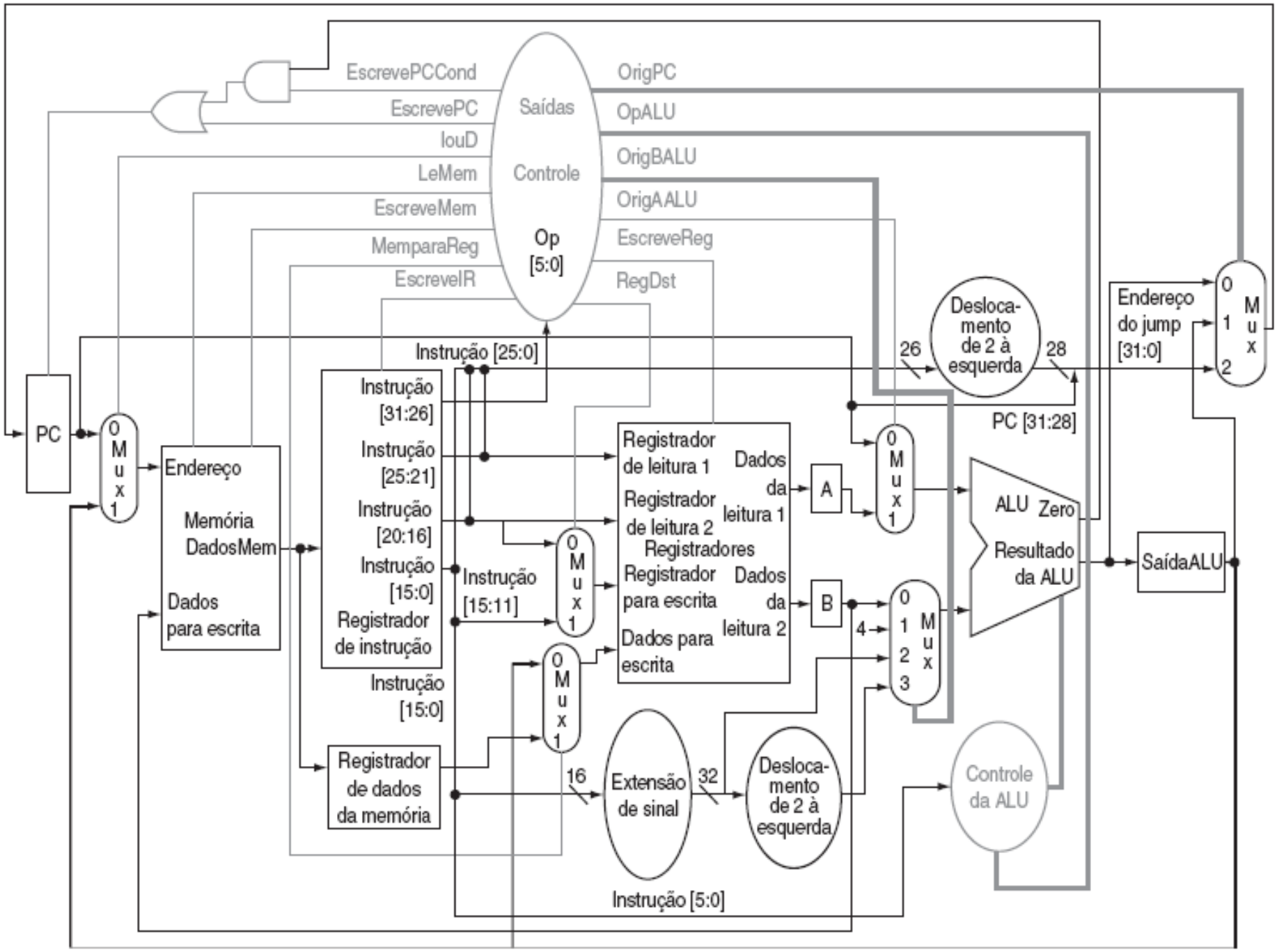
4.Conclusão de instrução tipo R ou acesso à memória

- Durante esta etapa, uma instrução *load* ou *store* acessa a memória e uma instrução lógica ou aritmética escreve seu resultado.
- Referência à memória
  - Se a instrução é um *load*, uma palavra de dados é lida da memória e é escrita no MDR. Se a instrução é um *store*, então, os dados são escritos na memória. O endereço usado é o calculado na etapa anterior e armazenado em SaídaALU.

MDR <= Memória [SaídaALU];

Memória [SaídaALU] <= B;

RegDst	Escreve Reg	OrigAALU	LeMem	Escreve Mem	MemparaReg	IouD	EscrevelR	Escreve PC	Escreve PCCond	OpALU	OrigBALU	OrigPC
			1 (*load*)	1 (*store*)		1						


EscrevePCCond
EscrevePC
IouD
LeMem
EscreveMem
MemparaReg
EscreveIR
Saídas
Controle
Op
[5:0]
OrigPC
OpALU
OrigBALU
OrigAALU
EscreveReg
RegDst
Instrução [25:0]
26
Deslocamento de 2 à esquerda
28
PC [31:28]
Endereço do jump [31:0]
PC
Endereço
Memória
DadosMem
Dados para escrita
Instrução [31:26]
Instrução [25:21]
Instrução [20:16]
Instrução [15:0]
Registrador de instrução
Instrução [15:11]
Instrução [15:0]
Registrador de leitura 1
Registrador de leitura 2
Registradores
Registrador para escrita
Dados para escrita
Dados da leitura 1
Dados da leitura 2
A
B
4
ALU
Zero
Resultado da ALU
SaídaALU
Registrador de dados da memória
16
Extensão de sinal
32
Deslocamento de 2 à esquerda
Controle da ALU
Instrução [5:0]
M u x

Etapas de Execução

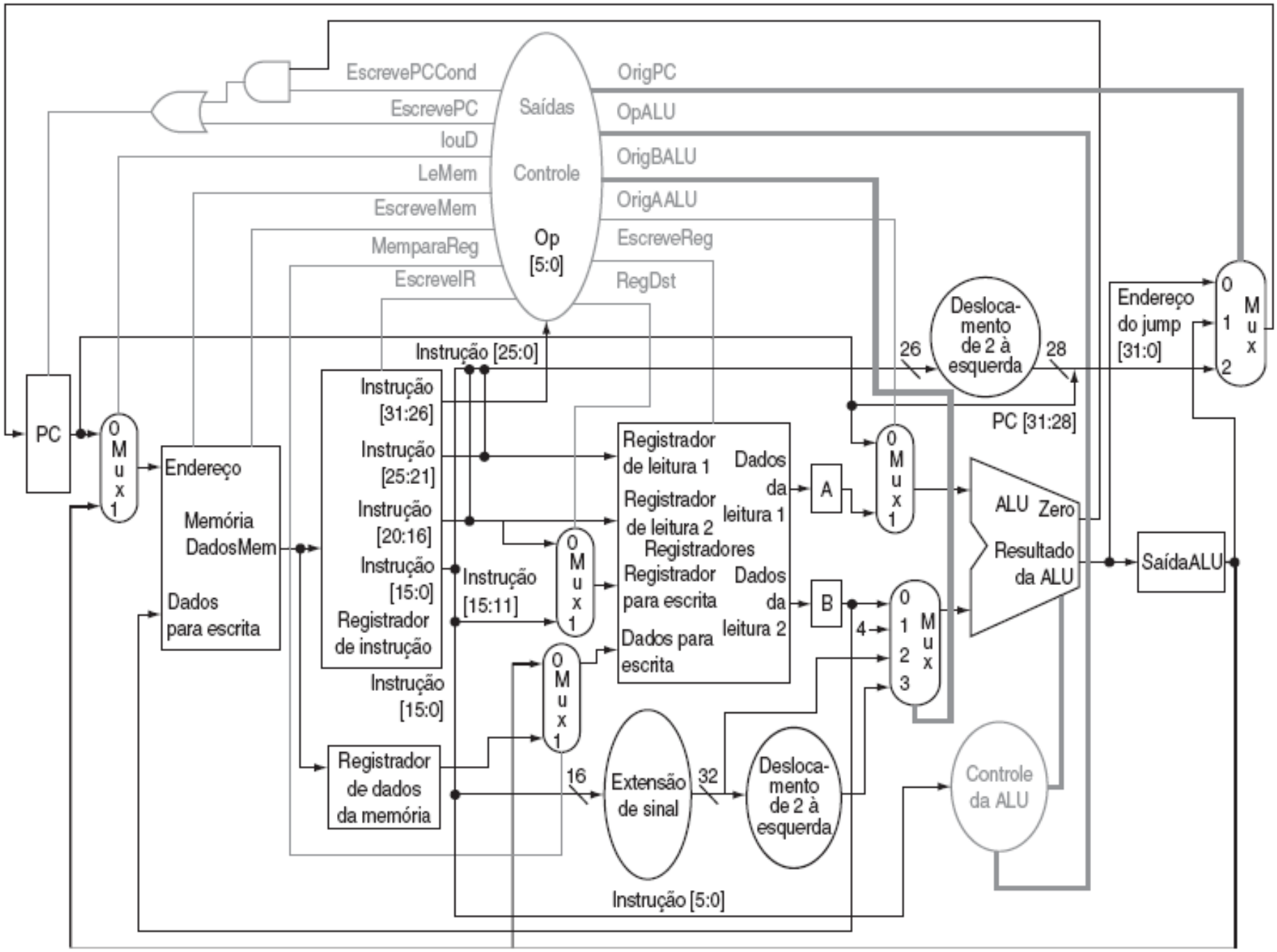
4.Conclusão de intrução tipo R ou acesso à memória

- Instrução do tipo R
  - Coloca o conteúdo de SaídaALU, que corresponde à saída da operação da ULA no ciclo anterior, no registrador de destino.

MDR <= Memória [SaídaALU];

Memória [SaídaALU] <= B;

RegDst	Escreve Reg	OrigAALU	LeMem	Escreve Mem	MemparaReg	IouD	EscrevelR	Escreve PC	Escreve PCCond	OpALU	OrigBALU	OrigPC
1	1				0							


EscrevePCCond
EscrevePC
IouD
LeMem
EscreveMem
MemparaReg
EscreveIR
Saídas
Controle
Op
[5:0]
OrigPC
OpALU
OrigBALU
OrigAALU
EscreveReg
RegDst
Instrução [25:0]
26
Deslocamento de 2 à esquerda
28
PC [31:28]
Endereço do jump [31:0]
PC
Endereço
Memória
DadosMem
Dados para escrita
Instrução [31:26]
Instrução [25:21]
Instrução [20:16]
Instrução [15:0]
Registrador de instrução
Instrução [15:11]
Instrução [15:0]
Registrador de leitura 1
Registrador de leitura 2
Registradores
Registrador para escrita
Dados para escrita
Dados da leitura 1
Dados da leitura 2
A
B
4
ALU
Zero
Resultado da ALU
SaídaALU
Registrador de dados da memória
16
Extensão de sinal
32
Deslocamento de 2 à esquerda
Controle da ALU
Instrução [5:0]
M u x

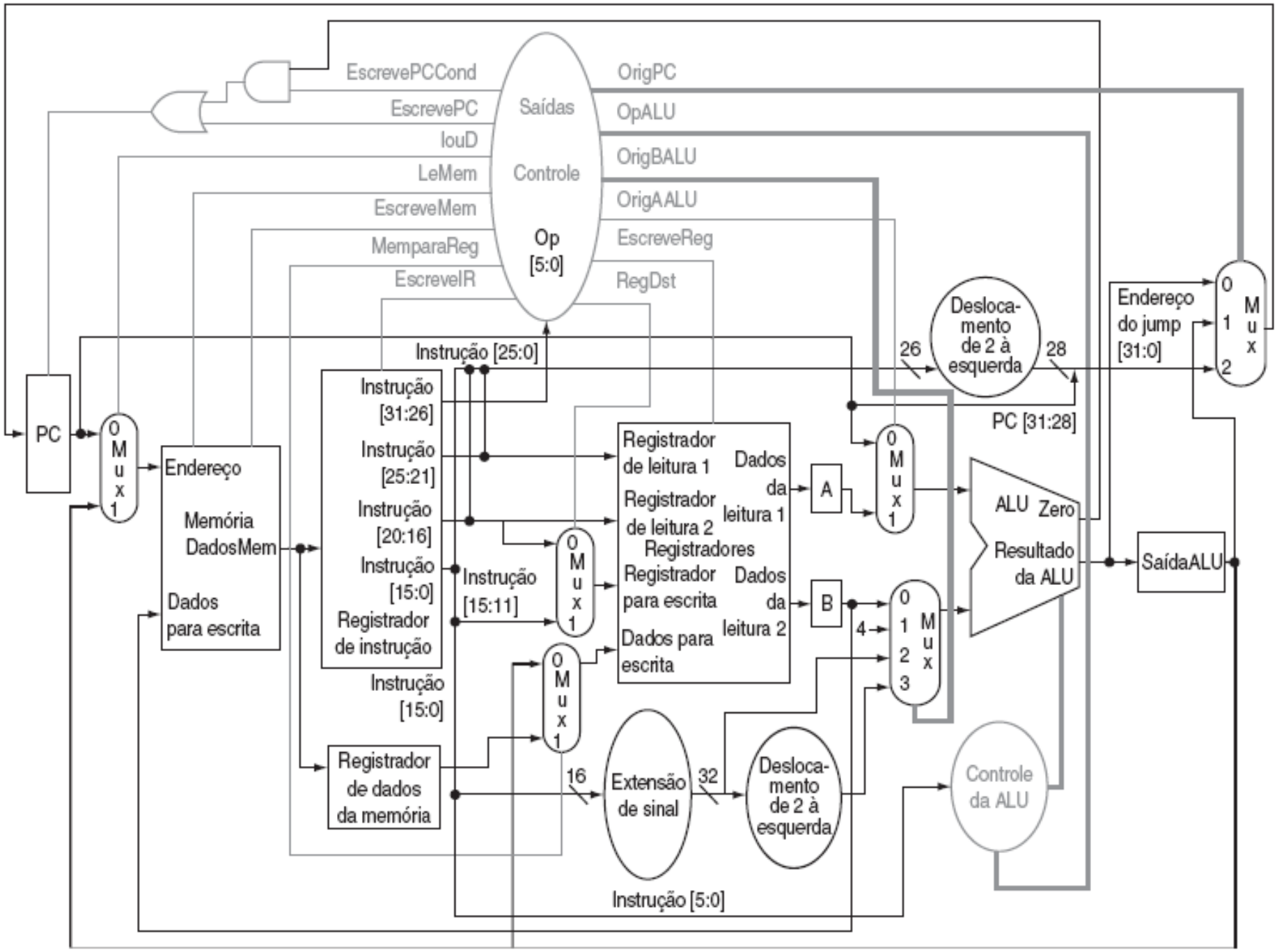
Etapas de Execução

5.Etapa de conclusão da leitura da memória

- Escrever os dados da leitura feita à memória, armazenados no MDR no ciclo anterior, no banco de registradores.

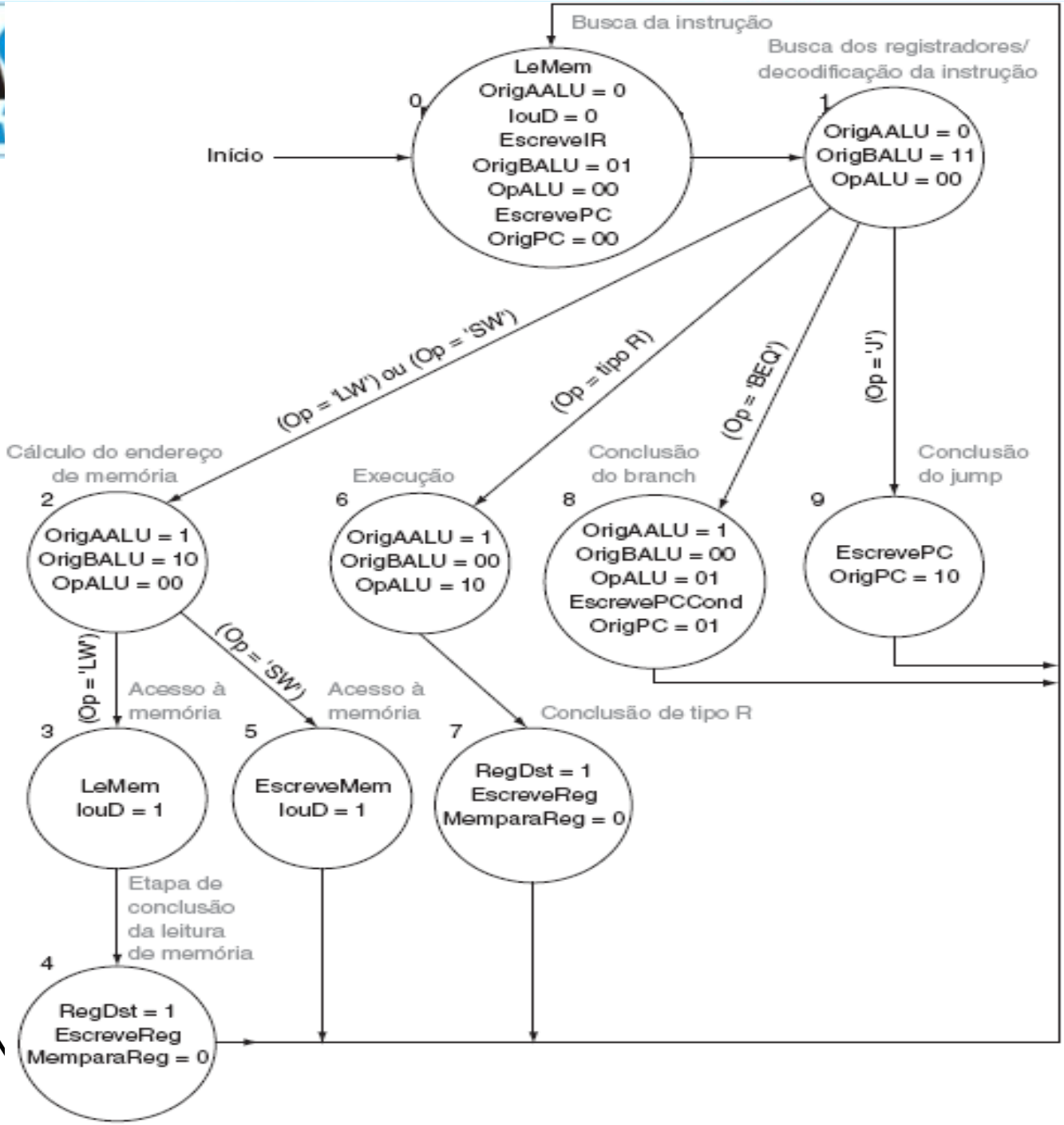
Reg [IR [20:16]] <= MDR;

RegDst	Escreve Reg	OrigAALU	LeMem	Escreve Mem	MemparaReg	IouD	EscrevelR	Escreve PC	Escreve PCCond	OpALU	OrigBALU	OrigPC
0	1				1							


EscrevePCCond
EscrevePC
IouD
LeMem
EscreveMem
MemparaReg
EscreveIR
Saídas
Controle
Op
[5:0]
OrigPC
OpALU
OrigBALU
OrigAALU
EscreveReg
RegDst
Instrução [25:0]
26
Deslocamento de 2 à esquerda
28
PC [31:28]
Endereço do jump [31:0]
PC
Endereço
Memória
DadosMem
Dados para escrita
Instrução [31:26]
Instrução [25:21]
Instrução [20:16]
Instrução [15:0]
Registrador de instrução
Instrução [15:11]
Instrução [15:0]
Registrador de leitura 1
Registrador de leitura 2
Registradores
Registrador para escrita
Dados para escrita
Dados da leitura 1
Dados da leitura 2
A
B
4
ALU
Zero
Resultado da ALU
SaídaALU
Registrador de dados da memória
16
Extensão de sinal
32
Deslocamento de 2 à esquerda
Controle da ALU
Instrução [5:0]
M u x

Etapas de Execução

Etapa	Ação para instruções tipo R	Ação para instruções de acesso à memória	Ação para desvios	Ação para jumps
Busca da instrução	IR <= Memória[PC] PC <= PC + 4			
Decodificação da instrução e busca dos registradores	A <= Reg[IR[25:21]] B <= Reg[IR[20:16]] SaídaALU <= PC + (estende-sinal (IR[15:0]) << 2)			
Execução, cálculo do endereço, conclusão do desvio/jump	SaídaALU <= A op B	SaídaALU <= A + estende-sinal (IR[15:0])	if (A == B) PC <= SaídaALU	PC <= {PC [31:28], (IR[25:0]],2'b00)}
Acesso à memória ou conclusão de instrução tipo R	Reg[IR[15:11]]<= SaídaALU	Load: MDR <= Memória[SaídaALU] ou Store: Memória [SaídaALU] <= B		
Conclusão da leitura da memória		Load: Reg[IR[20:16]] <= MDR		


Busca da instrução
Busca dos registradores/
decodificação da instrução
0
LeMem
OrigAALU = 0
IouD = 0
EscreveIR
OrigBALU = 01
OpALU = 00
EscrevePC
OrigPC = 00
Início
1
OrigAALU = 0
OrigBALU = 11
OpALU = 00
(Op = 'LW') ou (Op = 'SW')
(Op = tipo R)
(Op = 'BEQ')
(Op = 'J')
Cálculo do endereço
de memória
Execução
Conclusão
do branch
Conclusão
do jump
2
OrigAALU = 1
OrigBALU = 10
OpALU = 00
6
OrigAALU = 1
OrigBALU = 00
OpALU = 10
8
OrigAALU = 1
OrigBALU = 00
OpALU = 01
EscrevePCCond
OrigPC = 01
9
EscrevePC
OrigPC = 10
(Op = 'LW')
(Op = 'SW')
Acesso à
memória
Acesso à
memória
Conclusão de tipo R
3
LeMem
IouD = 1
5
EscreveMem
IouD = 1
7
RegDst = 1
EscreveReg
MemparaReg = 0
Etapa de
conclusão
da leitura
de memória
4
RegDst = 1
EscreveReg
MemparaReg = 0

Etapas de Execução

```
lw $t2, 0($t3)
lw $t3, 4($t3)
beq $t2, $t3, Label        #considere not
add $t5, $t2, $t3
sw $t5, 8($t3)
Label: ...
```