Arquitetura e Organização de Computadores II
Arquitetura MIPS

Prof. Dr. Daniel S. Marcon
danielstefani@unisinos.br

UNISINOS

Histórico

- 1981 - J. Hennessy e sua equipe de Stanford propõem a arquitetura MIPS
- 1984 - Hennessy funda a MIPS Computer Systems
- 1991 - A Silicon Graphics compra a MIPS
- 1999 - Um em cada três processadores embarcados possui arquitetura MIPS
- 2006 - É lançado o primeiro MIPS Multithread

Introdução

- Microprocessador Sem Estágios Interligados de Pipeline (Microprocessor Without Interlocked Pipeline Stages – MIPS)
- Processadores MIPS são do tipo RISC
  - Existe um conjunto bastante pequeno de instruções que o processador sabe executar
  - Pode-se criar todas as demais operações combinando-se instruções desse pequeno conjunto
  - Ex.: add, sub, mov, and, shift, compare equal/not equal, branch, jump, call, return
  - Considerado bastante didático
- Usado em diversos sistemas, como roteadores Cisco, access points da Linksys e video-games (Nintendo 64, Playstation, Playstation 2 e Playstation Portable)

Organização da Memória

- Endereçada a byte
- Uma palavra tem 32 bits (4 bytes)
- As palavras são alinhadas: dados de 32 bits precisam iniciar em endereços múltiplos de 4

	Value of 3 low-order bits of byte address							
Width of object	0	1	2	3	4	5	6	7
1 byte (byte)	Aligned	Aligned	Aligned	Aligned	Aligned	Aligned	Aligned	Aligned
2 bytes (half word)	Aligned		Aligned		Aligned		Aligned	
2 bytes (half word)		Misaligned		Misaligned		Misaligned		Misaligned
4 bytes (word)	Aligned				Aligned			
4 bytes (word)		Misaligned				Misaligned		
4 bytes (word)			Misaligned				Misaligned	
4 bytes (word)				Misaligned				Misaligned
8 bytes (double word)	Aligned							
8 bytes (double word)		Misaligned						
8 bytes (double word)			Misaligned					
8 bytes (double word)				Misaligned				
8 bytes (double word)					Misaligned			
8 bytes (double word)						Misaligned		
8 bytes (double word)							Misaligned	
8 bytes (double word)								Misaligned

Alinhamento de memória: https://www.ibm.com/developerworks/library/pa-dalign/

UNISINOS

Registradores

- 32 registradores de propósito geral de 32 bits
  - $0 .. $31
  - Operações inteiras e de endereçamento
- $0 tem sempre valor 0
- $31 guarda endereço de retorno de sub-rotina
- 32 registradores de ponto flutuante de 32 bits (precisão simples)
  - $f0 .. $f31
  - Podem ser usados em pares para precisão dupla
  - Registradores Hi e Lo para uso em multiplicação e divisão

Registradores - Convenção

UNISINOS

Instruções

- Tamanho de 32 bits
- Usa registradores de propósito geral e uma arquitetura load/store
  - 32 registradores + 32 registradores para ponto flutuante
- Suporta endereçamentos de memória básicos
  - Registrador, imediato, base+índice (deslocamento), relativo ao PC e pseudo-absoluto
- Suporta dados de 8, 16 e 32 bits para inteiros
- Suporta ponto flutuantes de acordo com IEEE 754 (32 e 64 bits)
- Objetiva um conjunto de instruções minimalista
- Formato de instruções fixo: R, I, J

UNISINOS

Formato das Instruções

- op-code: operação básica da instrução
- rs: primeiro registrador de origem
- rt: segundo registrador de origem
- rd: registrador de destino
- shamt: deslocamento (shift amount)
- funct: indica a função específica que a ULA irá realizar

Formato das Instruções

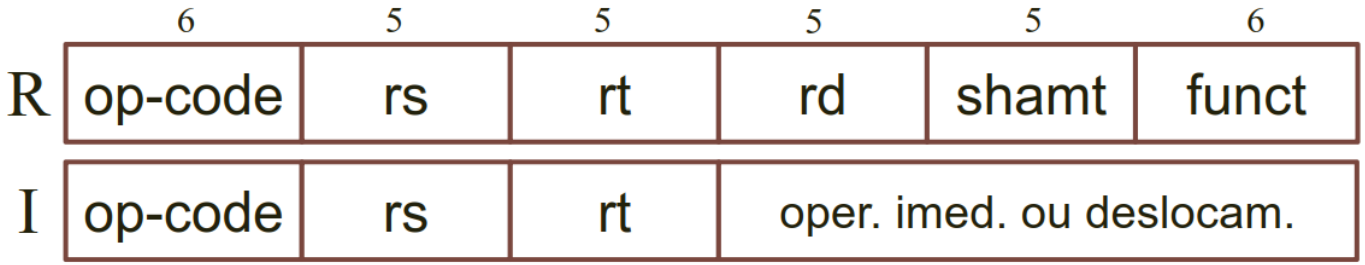
- Instruções do tipo R
  - Aritméticas e lógicas (add, sub, and, or)
  - Movimentação entre registradores
  - Exemplo: add $s1, $s2, $s3

UNISINOS

Formato das Instruções

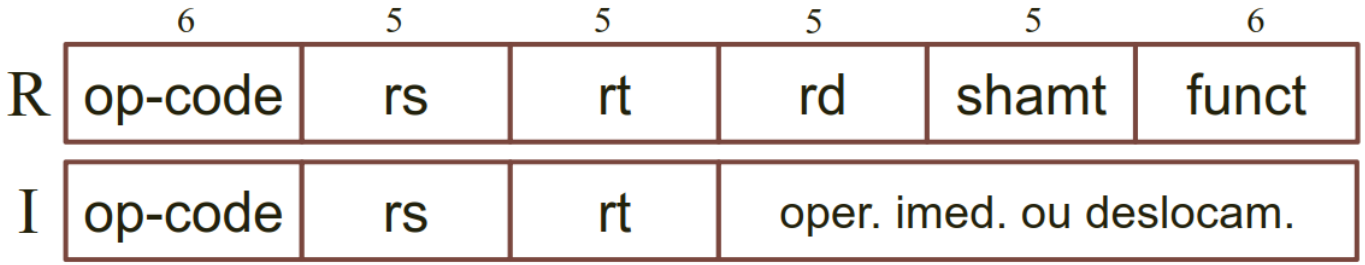
- Instruções do tipo I
  - load, store
  - Operações aritméticas e lógicas com operando imediato
  - Desvios condicionais (branches)
  - Exemplo 1: lw $s1, deslocam($s2)
  - Exemplo 2: sw $s1, deslocam($s2)
  - Exemplo 3: beq $s1, $s2, deslocam
  - Em branches: PC = PC + sign-ext$_{32}$(IR$_{15-0}$::00)

Formato das Instruções



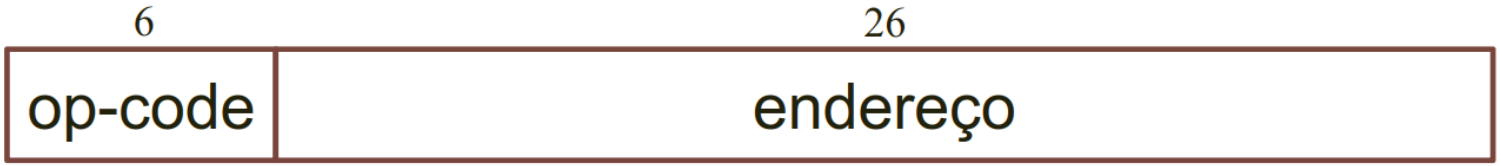
- Primeiros 3 campos são os mesmos do R
  - Facilita para o hardware
- Porém, o campo rt muda de significado

Formato das Instruções



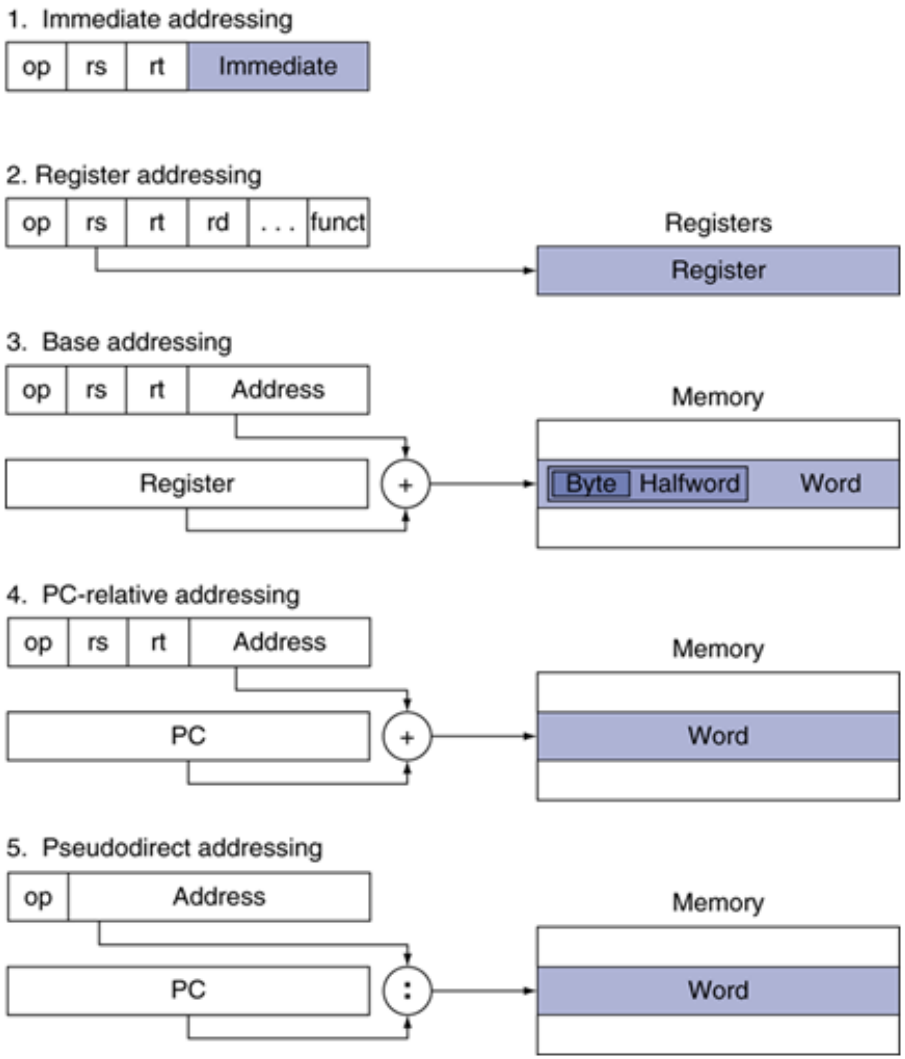
- Primeiros 3 campos são os mesmos do R
  - Facilita para o hardware
- Porém, o campo rt muda de significado
  - Formato R: rt é a 2a origem
  - Formato I: rt pode ser destino

Formato das Instruções



- Instruções do tipo J
  - Desvios com endereçamento absoluto
  - Chamada de sub-rotina
  - Exemplo: j endereço
- Em jumps:
  - PC = PC_{31-28}::IR_{25-0}::00)

Modos de Endereçamento

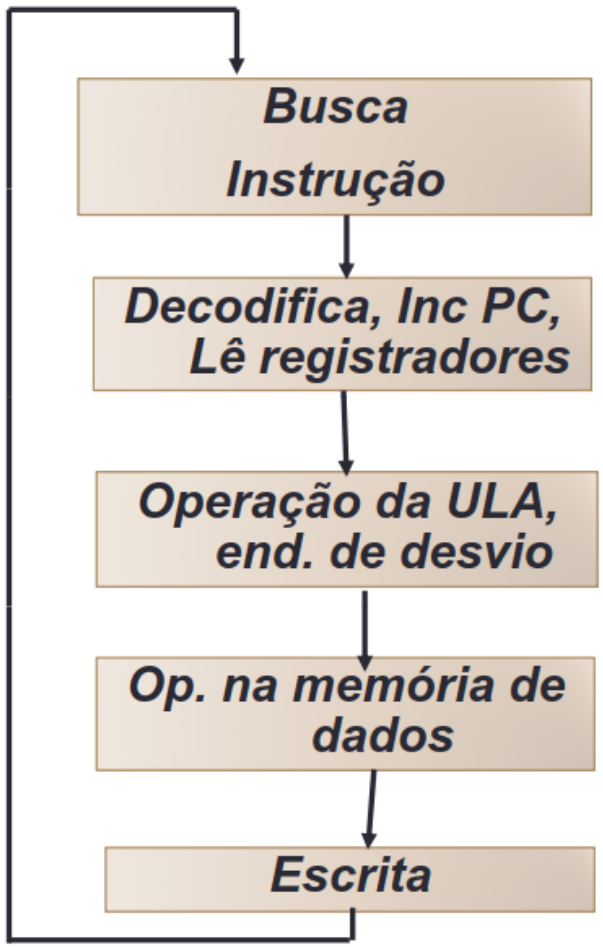


UNISINOS

Bloco Operacional Monociclo

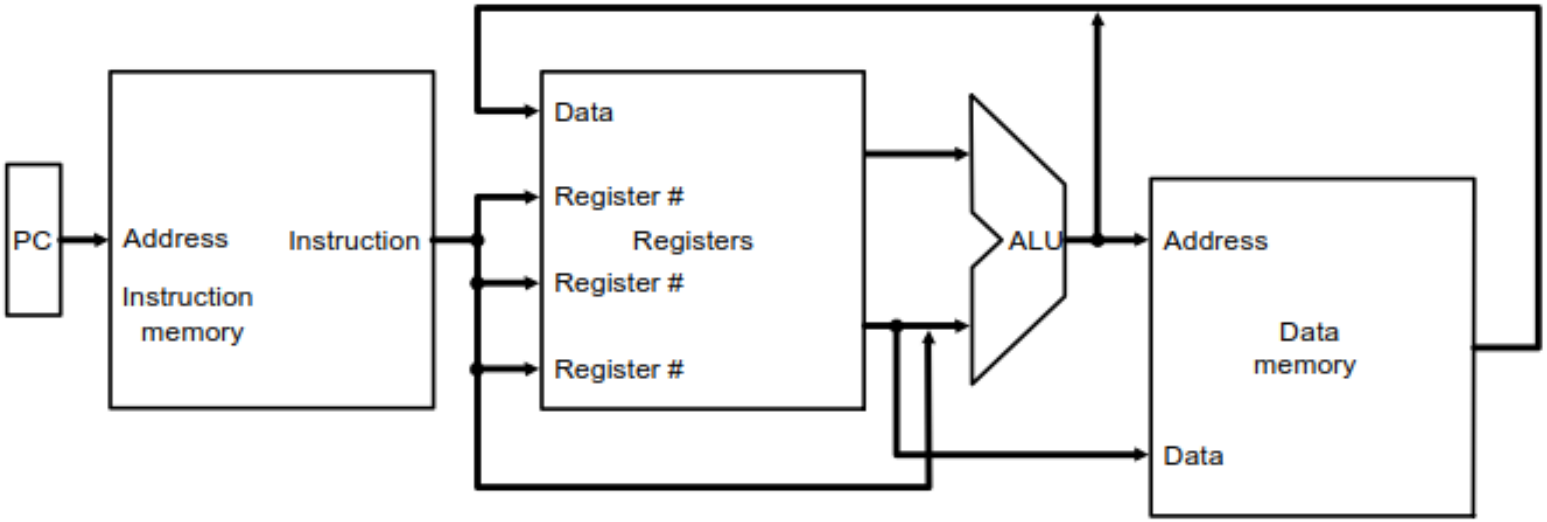
- Cada instrução é executada em um 1 ciclo de clock (CPI=1)
- Ciclo de clock deve ser longo o suficiente para executar a instrução mais longa

Passos para a Execução de uma Instrução



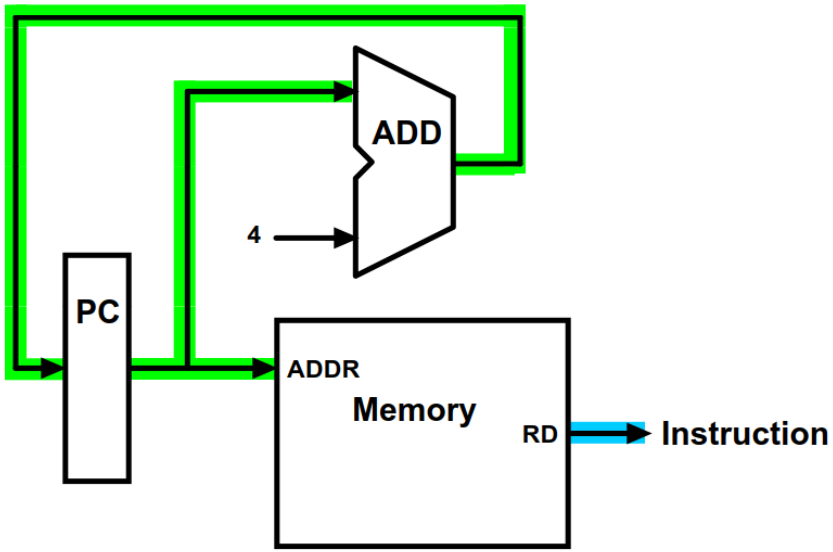
UNISINOS

Passos para a Execução de uma Instrução

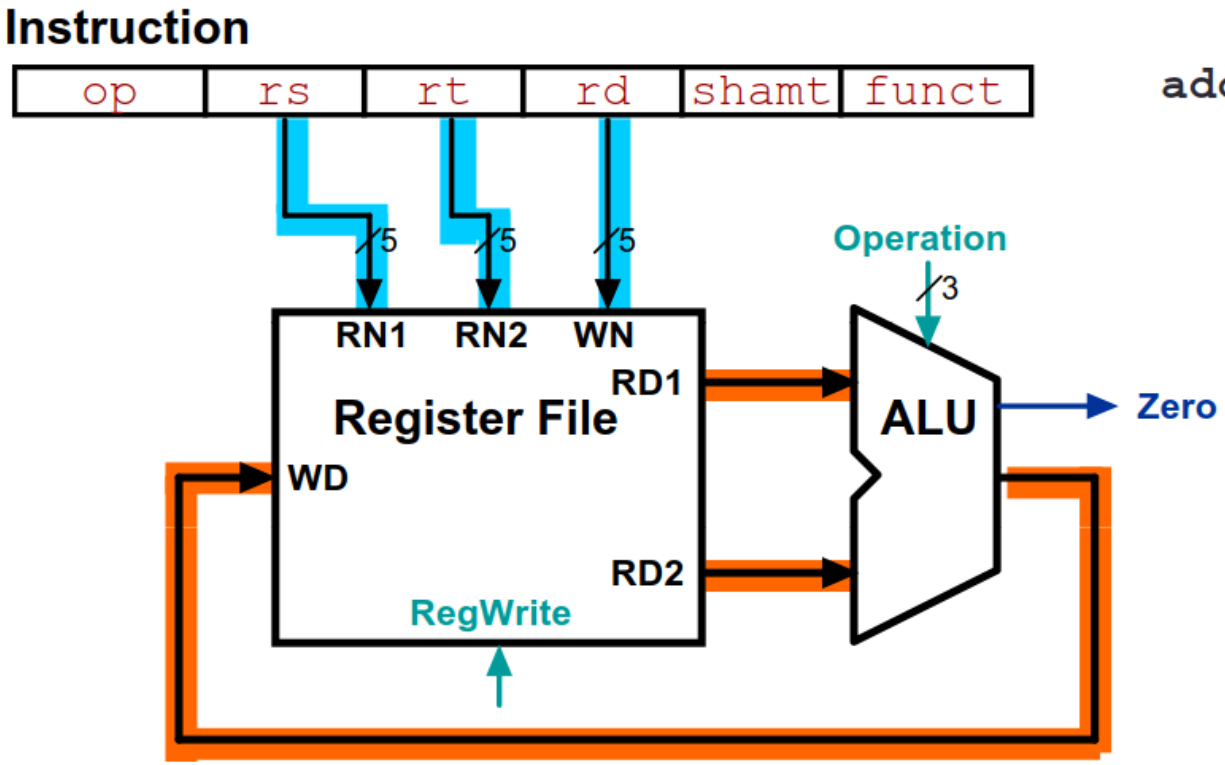


Busca da Instrução

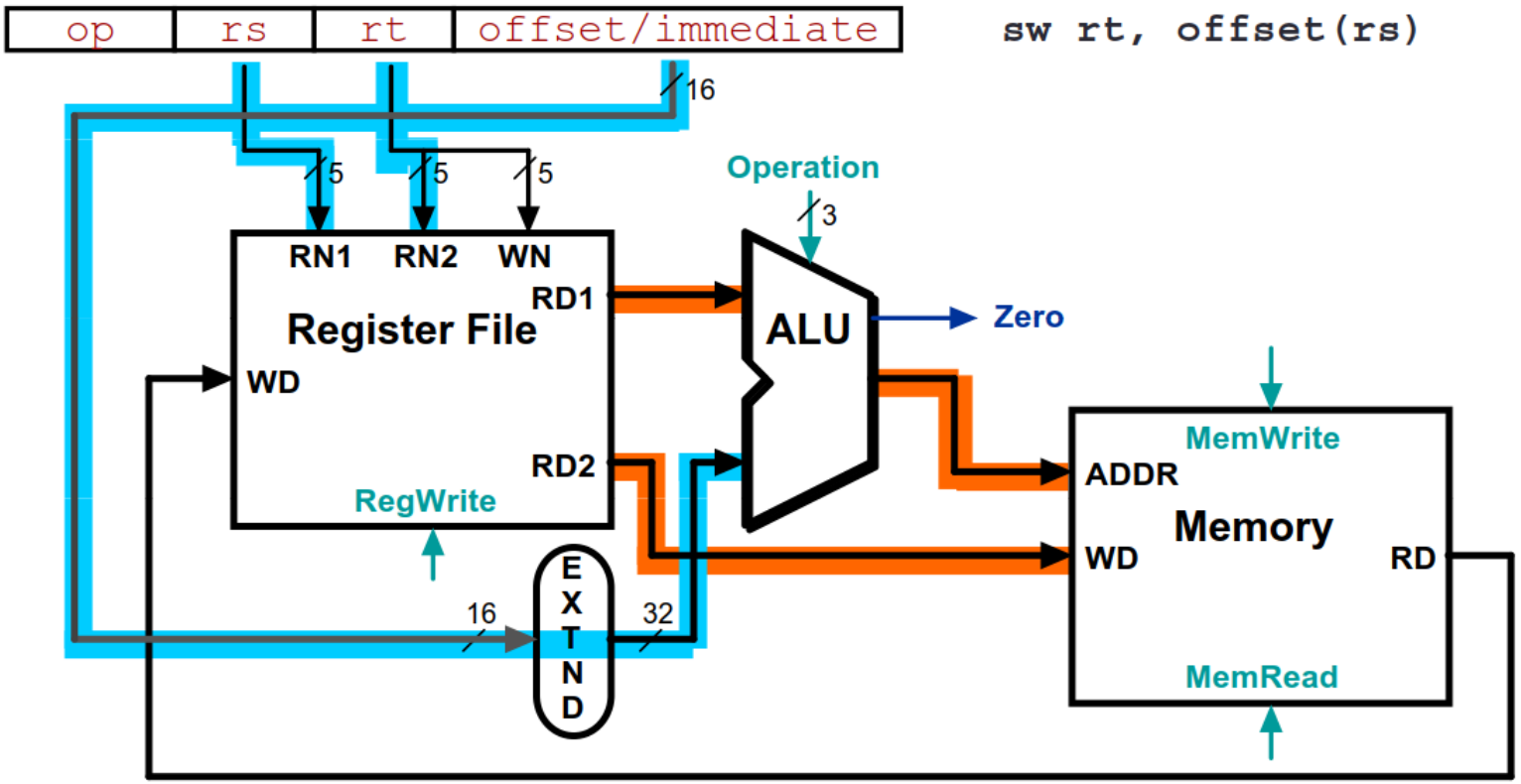
- O contador de programa contém o endereço da instrução em execução
- O endereço da próxima instrução é obtido pela soma de 4 posições ao contador de programa
- A instrução lida é usada por outras porções da parte operativa



Instruções Aritméticas

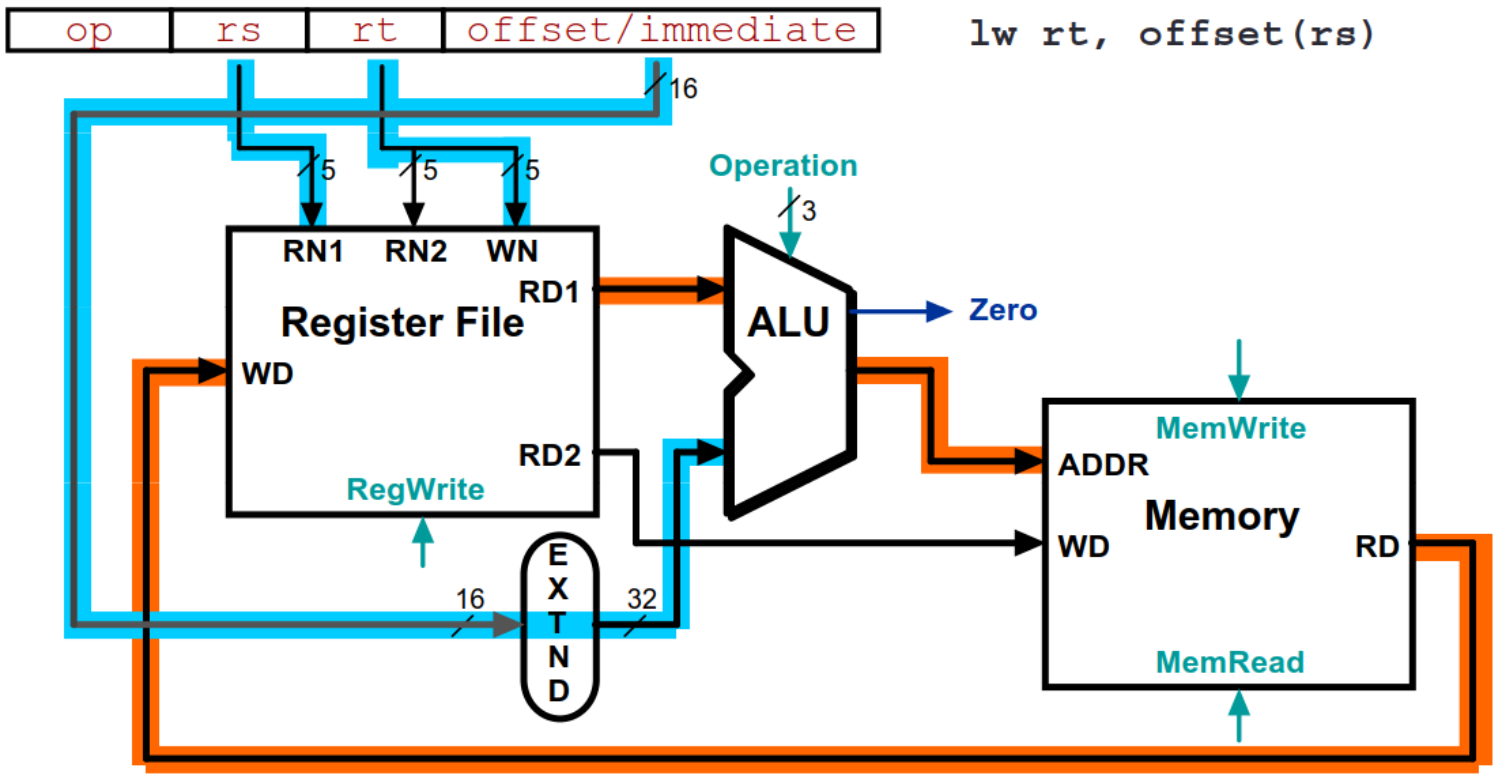


Escrita na Memória



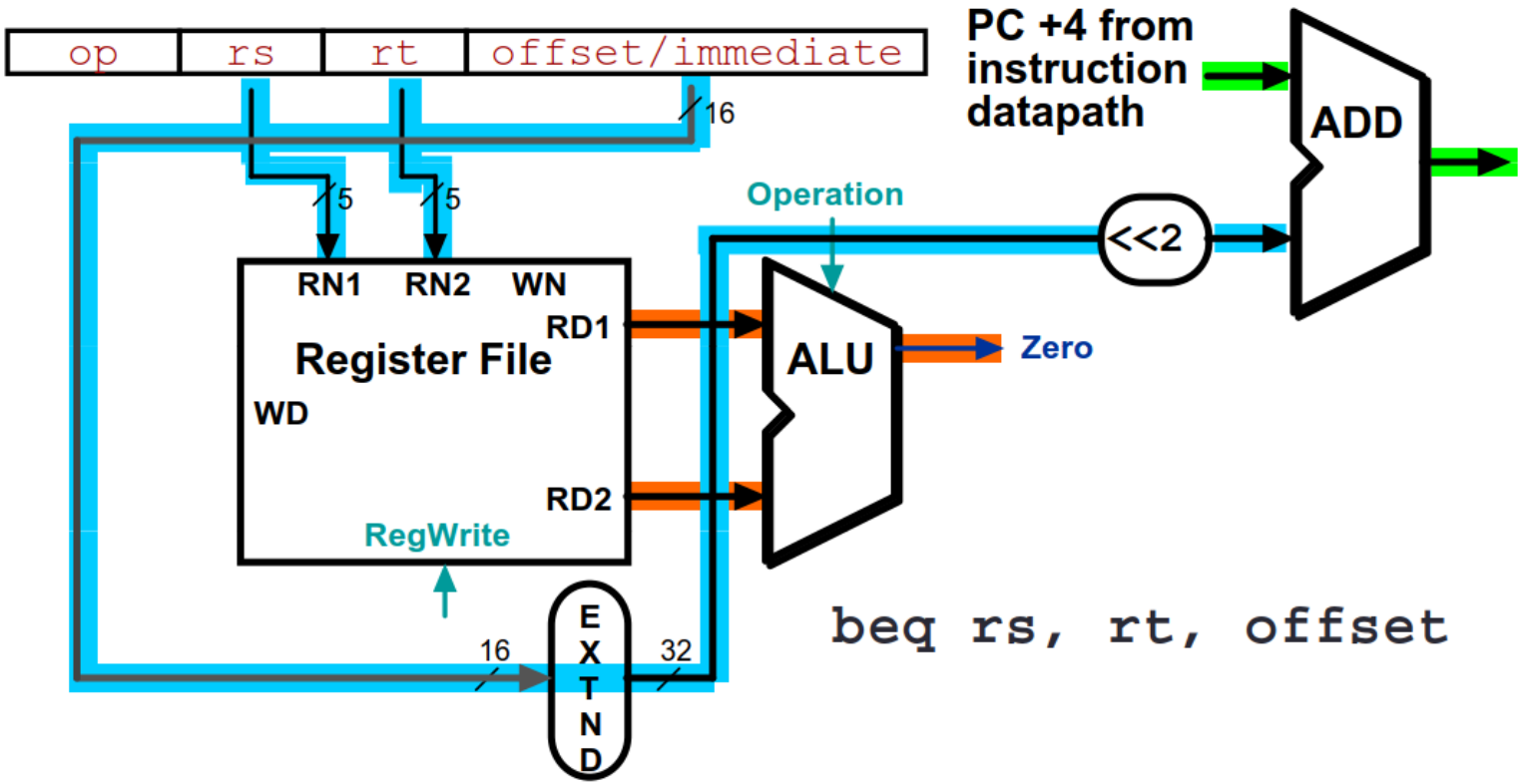
UNISINOS

Leitura da Memória



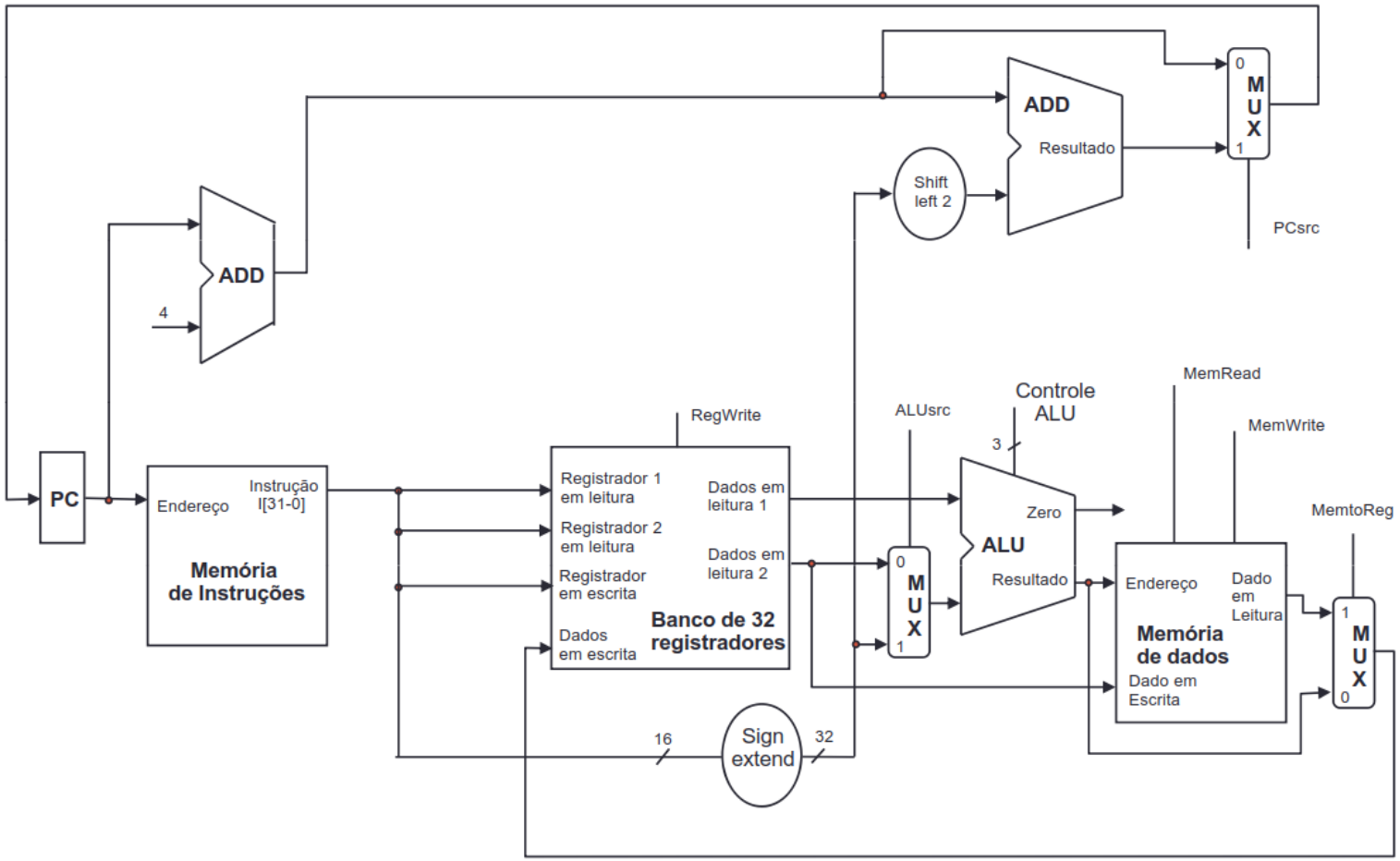
UNISINOS

Desvio Condicional



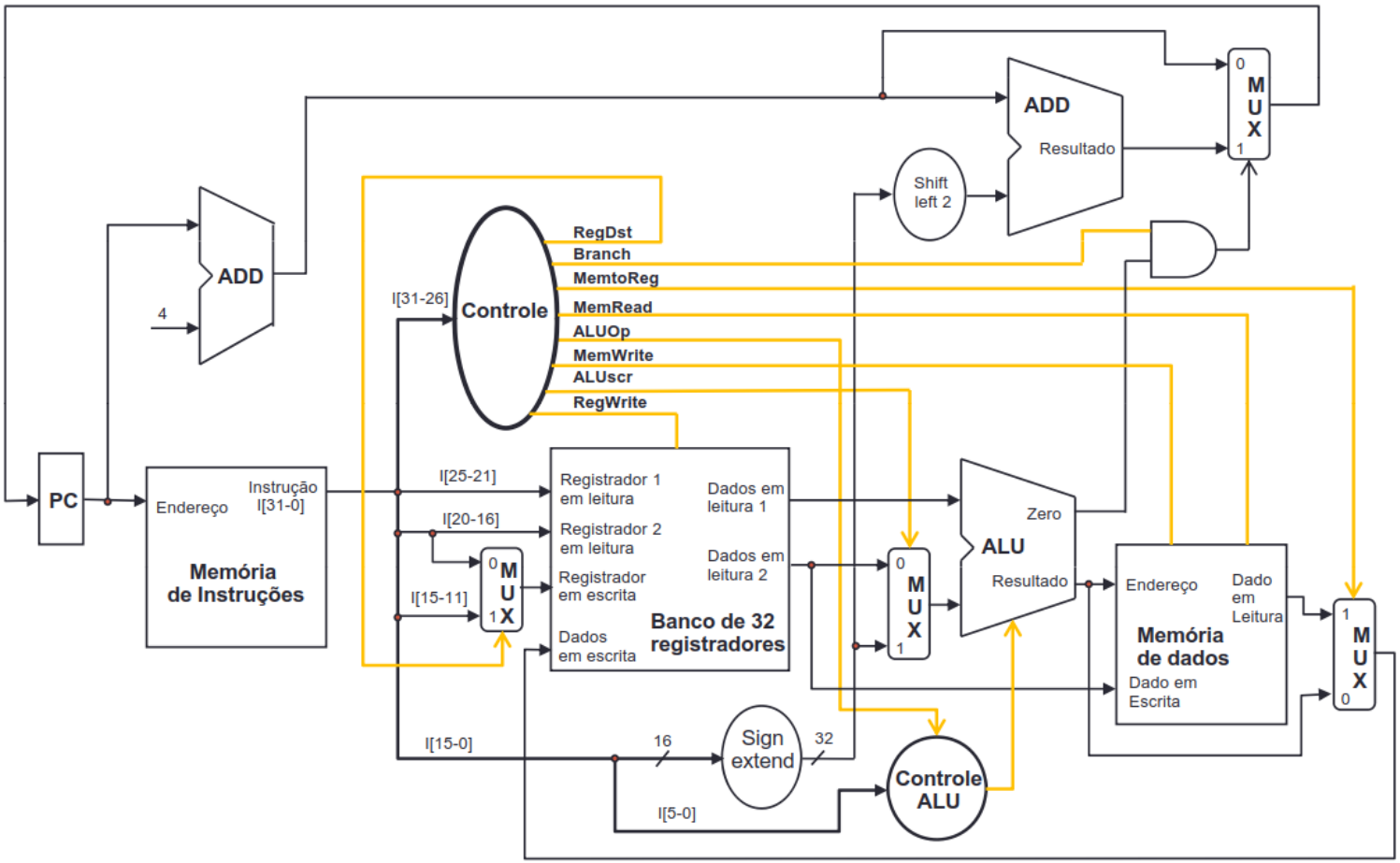
UNISINOS

Bloco Operacional Completo



UNISINOS

Bloco Operacional Completo com Controle



UNISINOS

Problemas com o Monociclo

UNISINOS

Problemas com o Monociclo

- Monociclo aumenta o tempo de ciclo (reduz a frequência) para caber a instrução mais lenta (ineficiência!)
- **Desempenho**: velocidade limitada pelo pior caso (instrução mais lenta)
- Supondo os seguintes atrasos: memórias (4 ns), ULA (2 ns), banco de registradores (1 ns) e somadores (1 ns)

instrução	busca	Lê registradores	Cálculo na ULA	Acessa memória de dados	Escreve em registrador	Total
add, sub, and, or	4ns	1ns	2ns	--	1ns	8ns
beq	4ns	1ns	2ns	--	--	7ns
sw	4ns	1ns	2ns	4ns	--	11ns
lw	4ns	1ns	2ns	4ns	1ns	12ns

UNISINOS

Alternativas

UNISINOS

Alternativas

- Fazer clock com frequência variável
  - Muito custoso e complicado

Alternativas

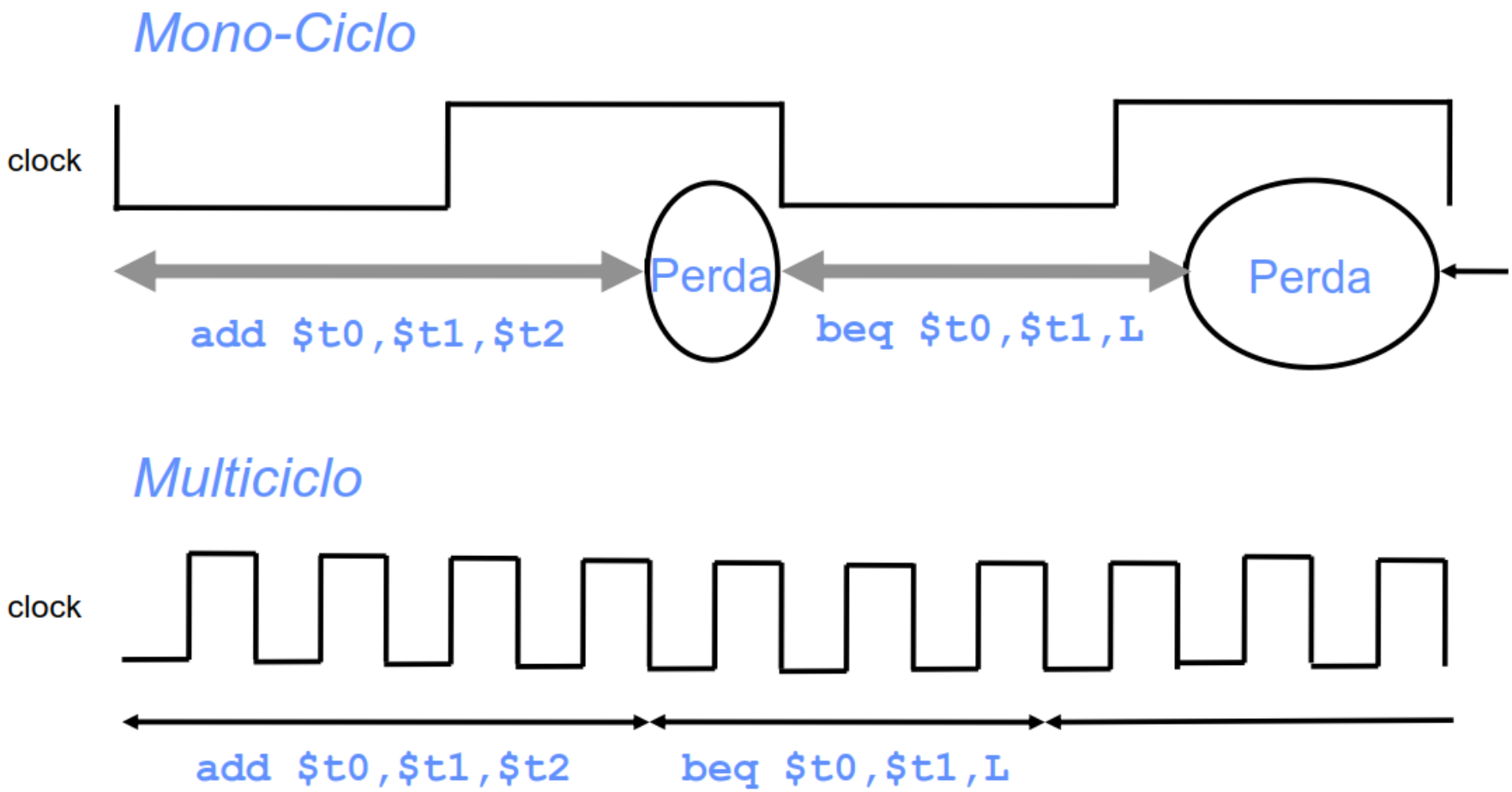
- Fazer clock com frequência variável
  - Muito custoso e complicado
- Dividir a instrução em pequenas funções e executá-las em vários ciclos de clock, uma (ou mais) por ciclo

Bloco Operacional Multiciclo

- Quebra o ciclo de execução em vários passos (como?)
- Executa cada passo em um ciclo de clock
- Vantagens
  - Ciclo limitado pela operação mais lenta e não pela instrução mais lenta
  - Instruções mais simples executam mais rapidamente (ou seja, em menos ciclos)
  - Maior desempenho no final
  - Versátil (pode ser mais facilmente estendido com outras instruções)

UNISINOS

Clock: Monociclo vs. Multiciclo



UNISINOS

Como dividir o processador em partes?

- Inserindo registradores
  - Quando valor é computado em um ciclo e utilizado em outro ciclo
  - Quando entradas de unidade funcional podem mudar antes que a saída seja salva em outro registrador ou memória
  - Exemplo: Instruction Register
    - Memória vai mudar saída devido à atualização do PC e campos da instrução precisam se manter estáveis nas entradas do banco de registradores durante todos os ciclos

Resumidamente

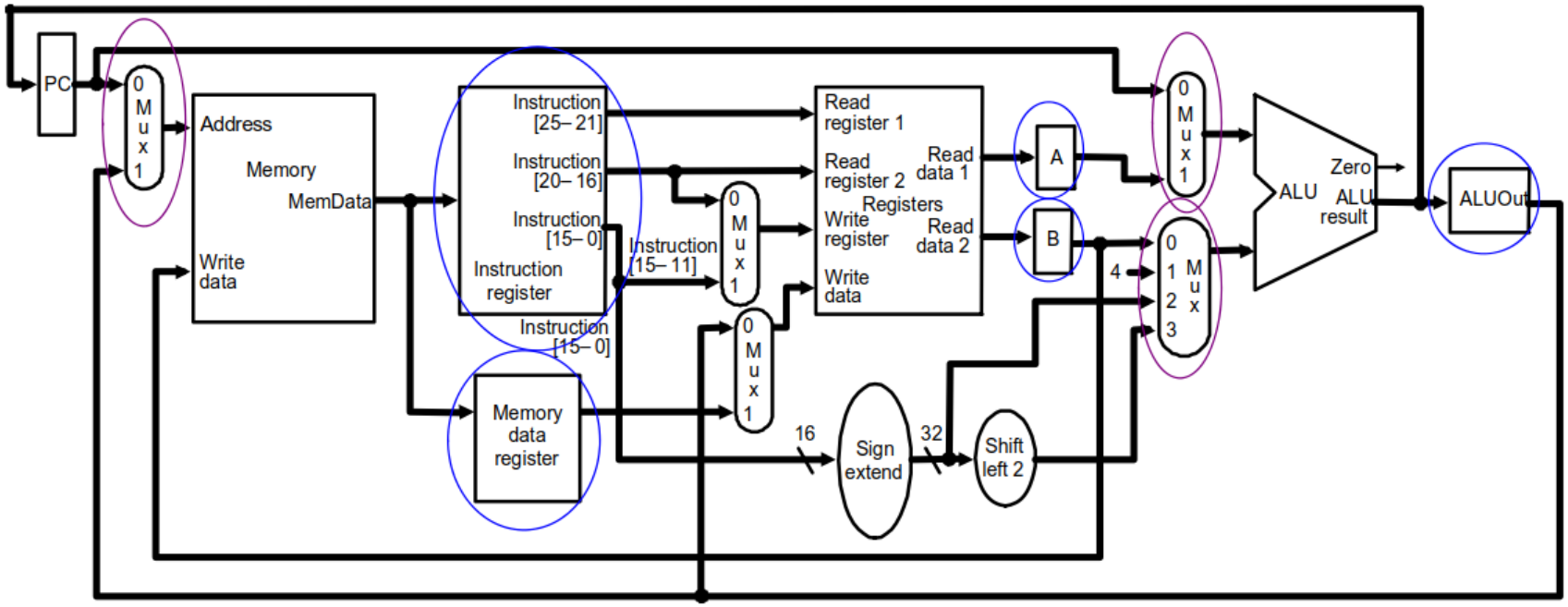
- Máquina monociclo
  - Todas as operações devem ser feitas em um só ciclo
  - Duração do ciclo calculada pelo pior caso
  - Leitura da instrução e acesso à memória no mesmo ciclo: duas memórias
  - Cálculos de endereço e operações aritméticas no mesmo ciclo: três unidades funcionais (ALU, somadores)
- Máquina multiciclo
  - Vários ciclos por instrução
  - Cada instrução pode ser executada em um número diferente de ciclos
  - Unidades funcionais podem ser reutilizadas em ciclos distintos
  - Pequeno acréscimo de multiplexadores e registradores
- Compromisso no desempenho (multiciclo)
  - CPI aumenta $\rightarrow$ desempenho cai
  - Período do relógio diminui $\rightarrow$ desempenho sobe

Passos das Instruções

1. Busca da instrução/Incremento do PC
2. .
   - Decodificação da instrução
   - Leitura dos registradores, mesmo que não sejam utilizados
   - Cálculo do endereço do branch, mesmo que instrução não seja branch
3. (operação depende do tipo)
   - Tipo R: execução da operação
   - Load e store: cálculo do endereço efetivo do operando
   - Branch: determina se deve ser executado e atualiza PC (**acaba**)
   - Jump: atualiza PC (**acaba**)
4. (operação depende do tipo)
   - Tipo R: grava resultado no registrador (**acaba**)
   - Store: grava dado na memória (**acaba**)
   - Load: busca dado na memória
5. Load: grava resultado no registrador (**acaba**)

UNISINOS

Bloco Operacional Multiciclo - Novos Componentes

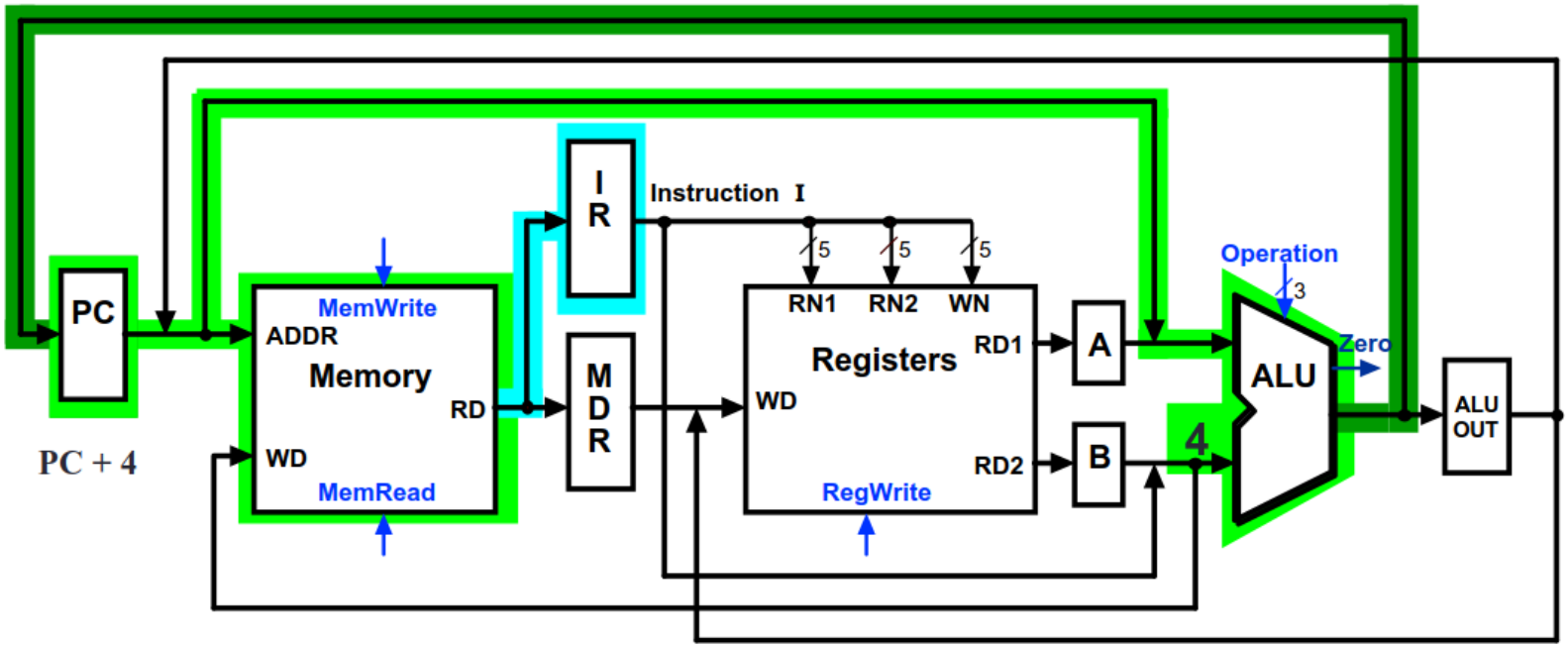


Novos registradores
Novos Multiplexadores

Bloco Operacional Multiciclo - Novos Componentes

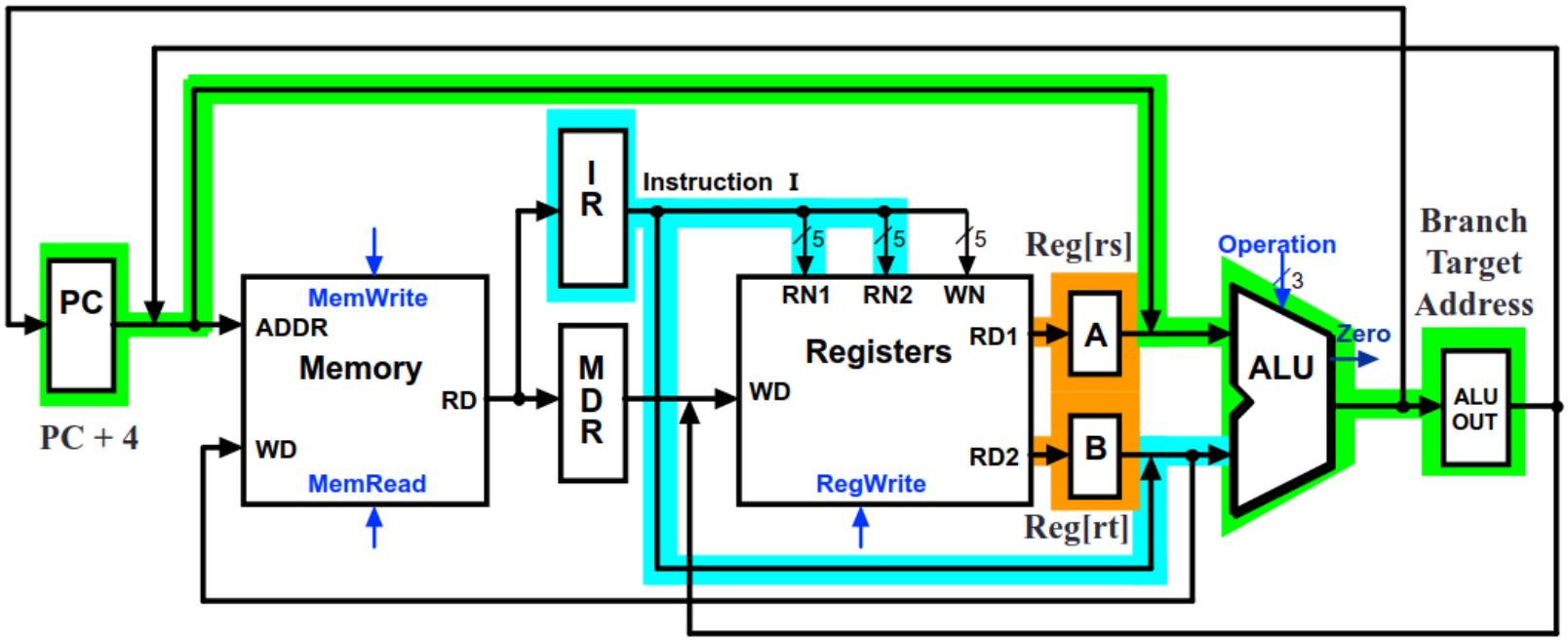
- Uma única memória para dados e instruções
- Uma única unidade lógica e aritmética para todas as operações
  - Operações das instruções do tipo R
  - Cálculo do PC = PC + 4
  - Cálculo de endereço efetivo de memória: base + deslocamento
  - Cálculo de endereço de desvio: PC + deslocamento
- Novos registradores
  - Registrador de Instruções
  - Registrador de Dados da Memória (MDR)
  - A e B: guardam valores dos operandos do banco de registradores
  - ALU out: guarda valor da saída da ULA
- Novos multiplexadores ou extensão dos já existentes

Passo 1 – Busca da Instrução



- IR = Memory[PC];
- PC = PC + 4;

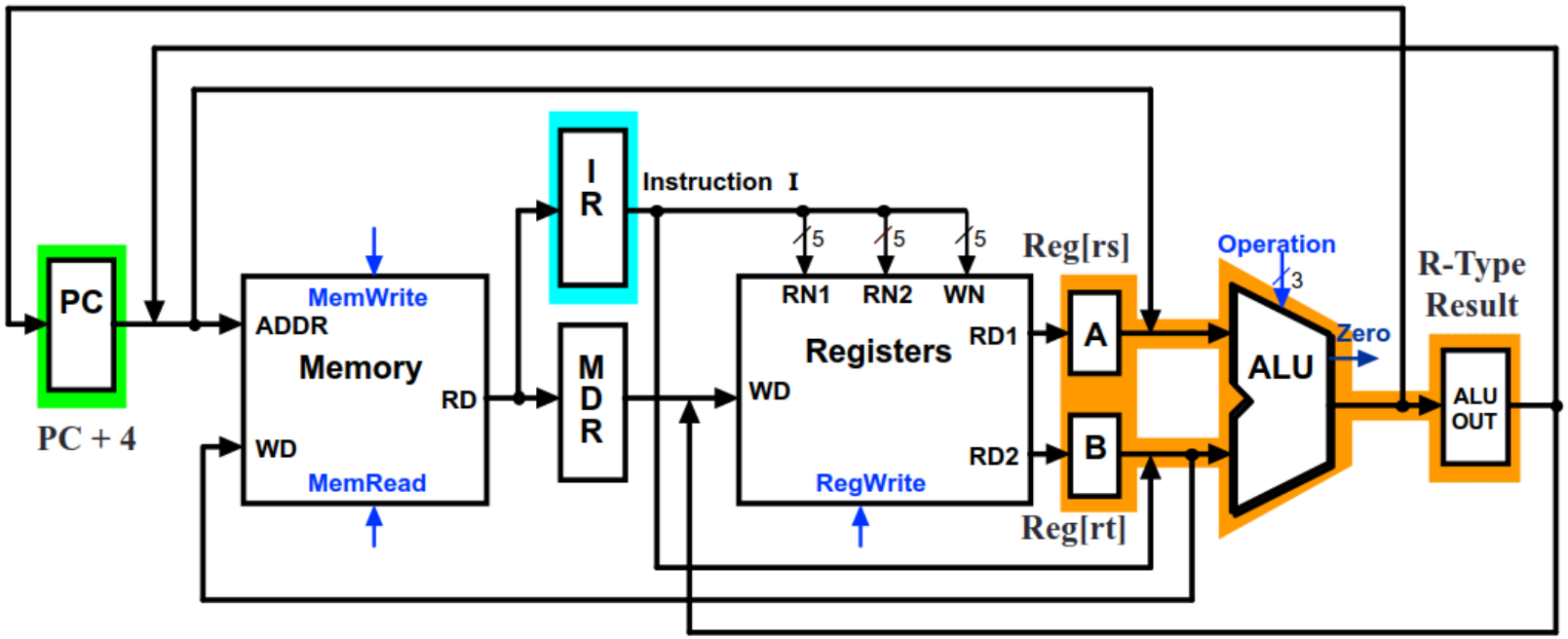
Passo 2 – Decodificação da Instrução / Busca nos Registradores



- A = Reg[IR[25-21]]; (A = Reg[rs])
- B = Reg[IR[20-15]]; (B = Reg[rt])
- ALUOut = (PC + sign-extend(IR[15-0]) « 2)

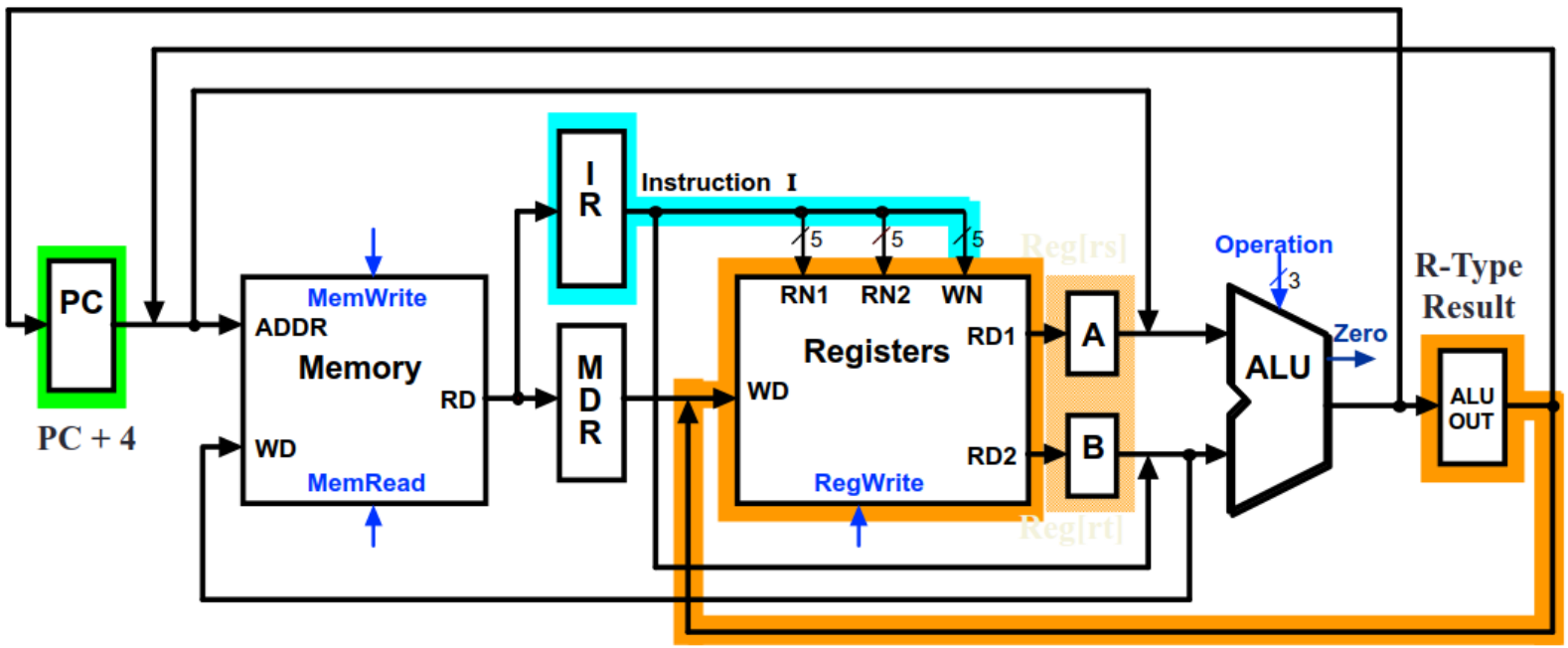
UNISINOS

Passo 3 – Instruções do Tipo R



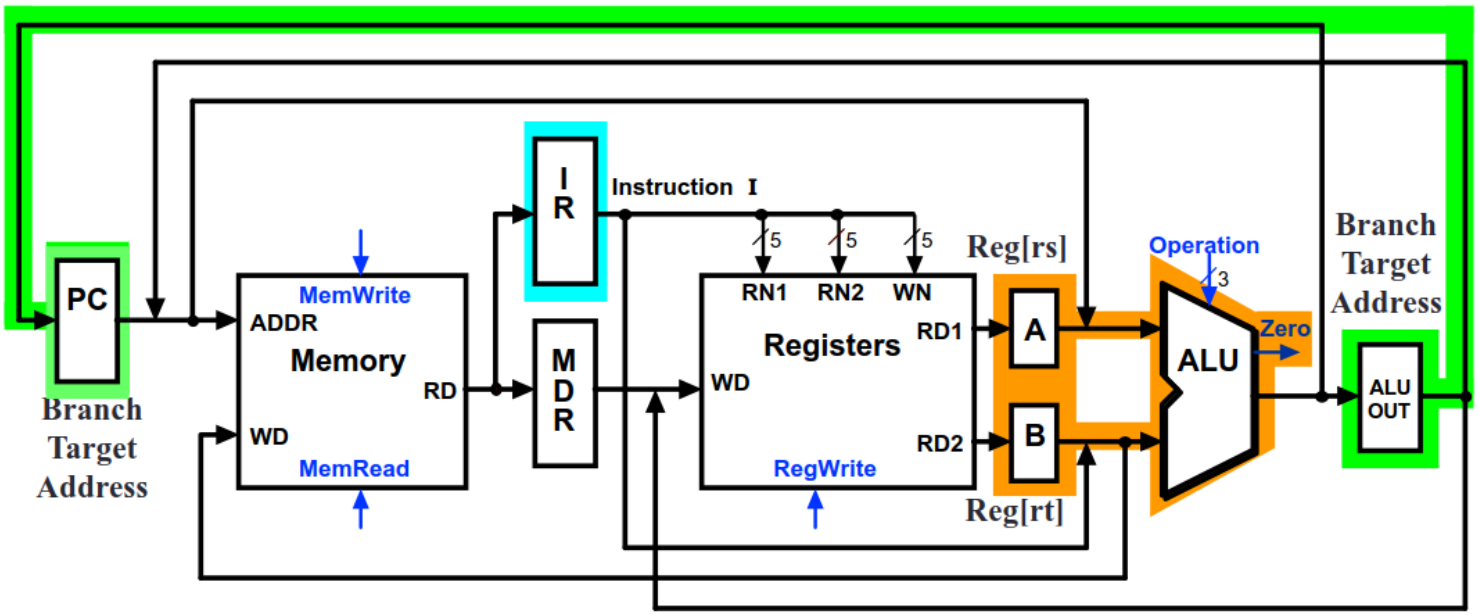
- ALUOut = A op B

Passo 4 – Instruções do Tipo R



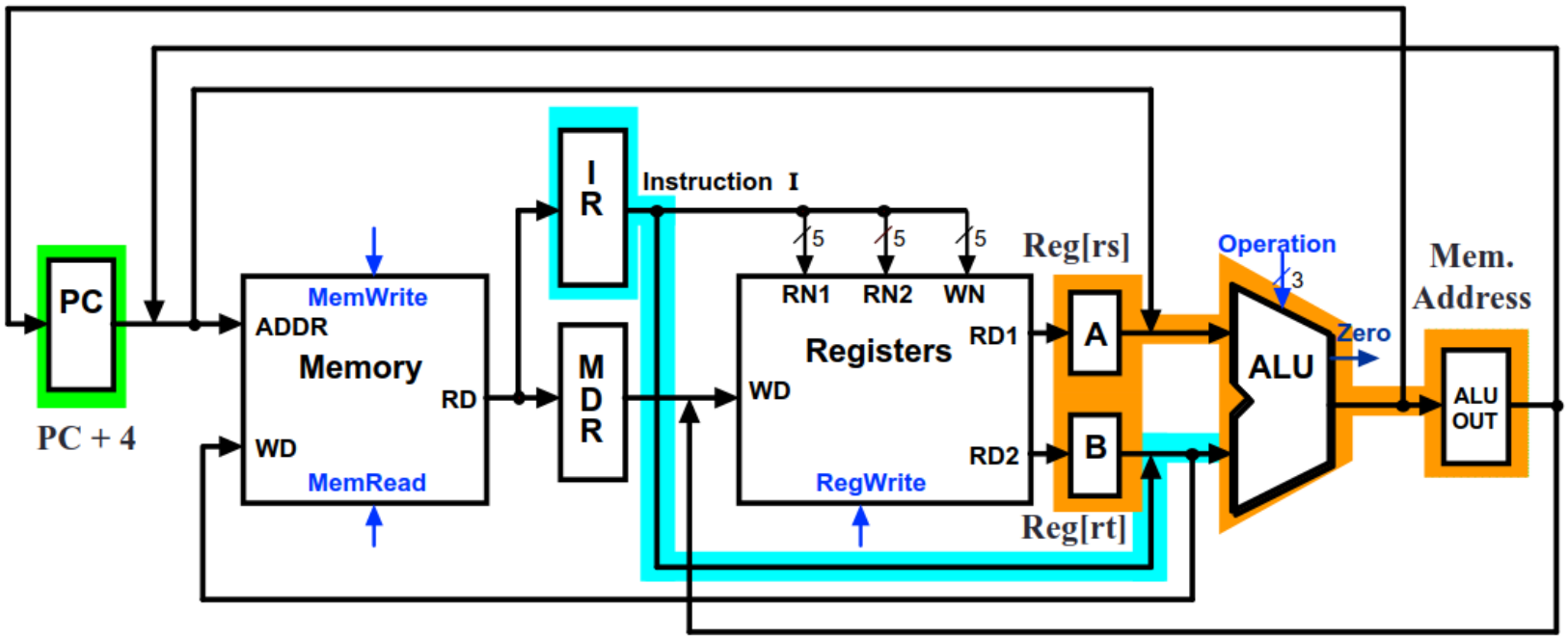
- Reg[IR[15:11]] = ALUOut

Passo 3 - Branch



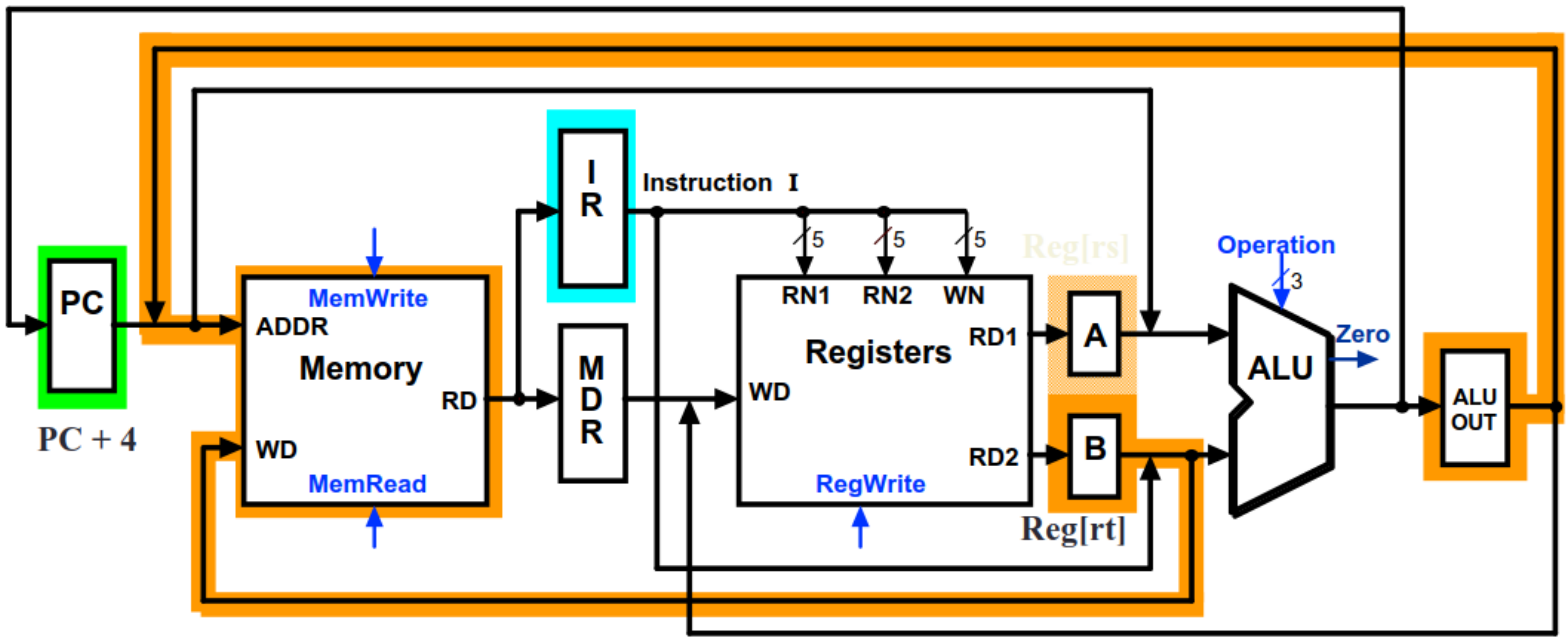
- if (A == B) PC = ALUOut;

Passo 3 – Instruções de Referência à Memória



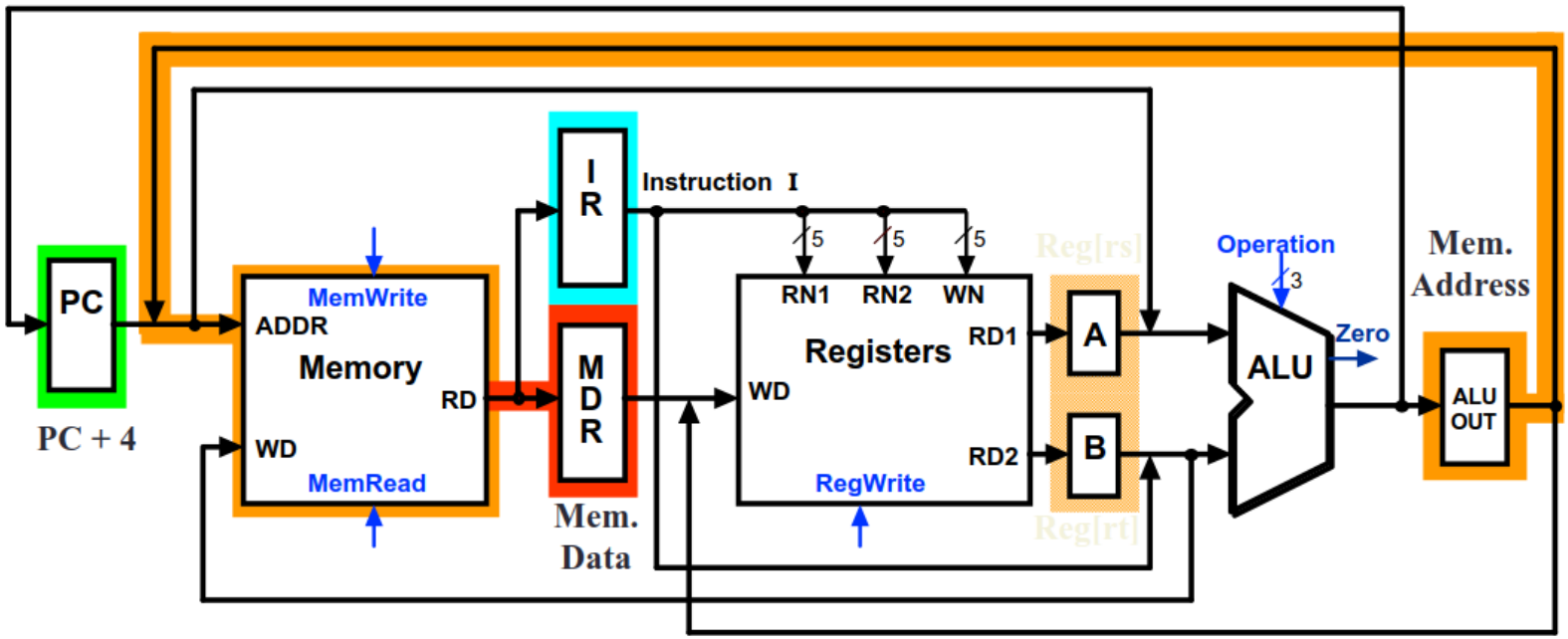
- ALUOut = A + sign-extend(IR[15-0]);

Passo 4 – Instruções de Referência à Memória - Store



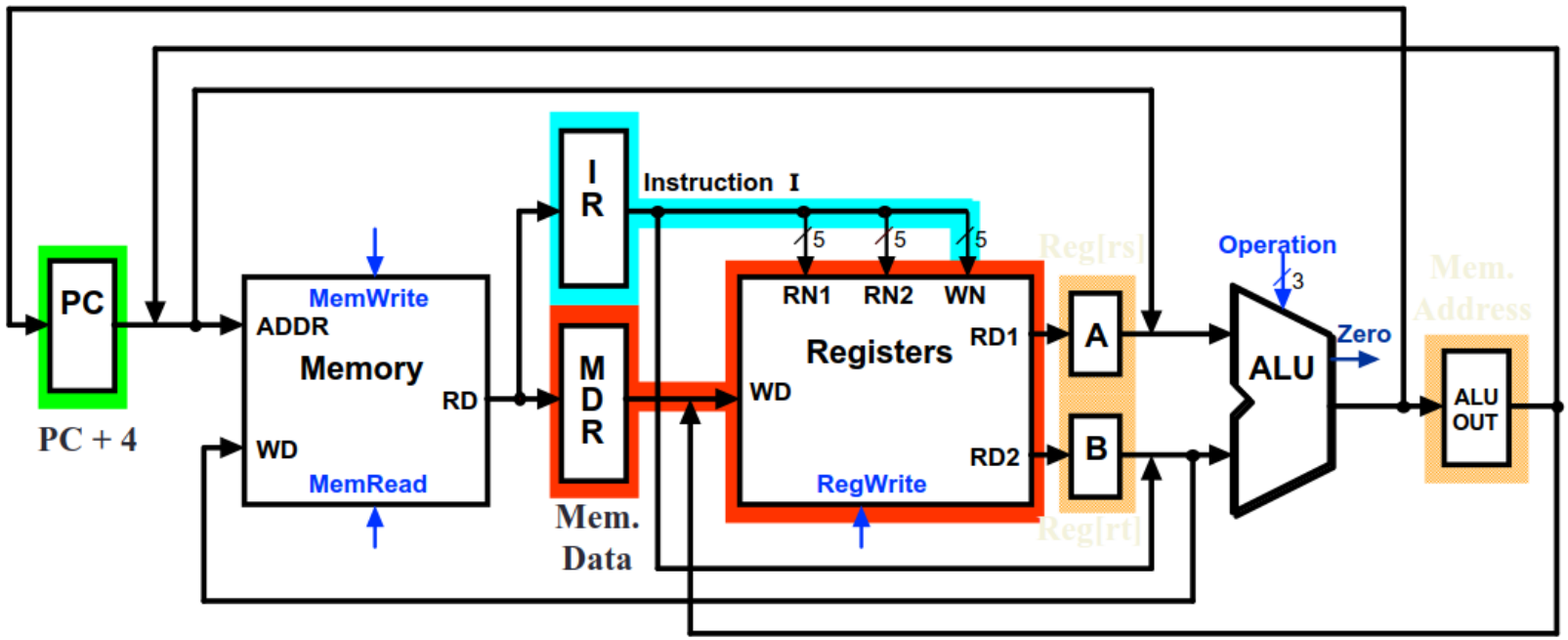
- Memory[ALUOut] = B;

Passo 4 – Instruções de Referência à Memória - Load



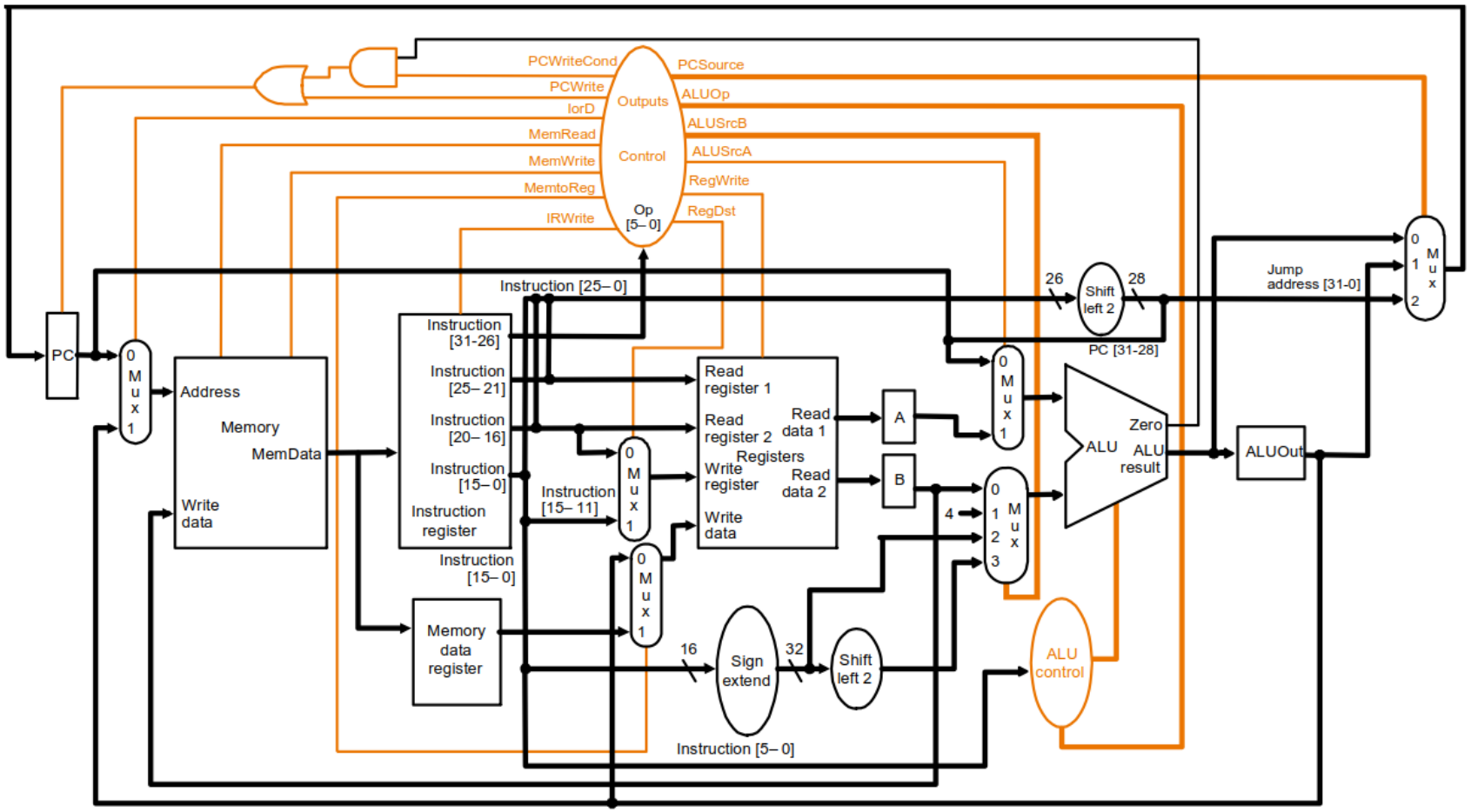
- MDR = Memory[ALUOut];

Passo 5 – Instruções de Referência à Memória - Load



- Reg[IR[20-16]] = MDR;

Bloco Operacional Multiciclo Completo



UNISINOS

Referências

HENNESSY, J.L. PATTERSON, D. A. Arquitetura de Computadores: uma abordagem quantitativa. Rio de Janeiro: Campus, 2003. 827p. ISBN: 85-352-110 85-35285-3.

UNISINOS