Universidade Federal do Rio Grande do Sul
Instituto de Informática

Organização de Computadores

Aula 06

Bloco operacional – versão multi-ciclo

Fim da Aula anterior: Ineficiência do Projeto Mono-Ciclo

- **Apesar de todas instruções serem executadas num ciclo de relógio e funcionar corretamente, é muito ineficiente,**
- **Uma das razões é que o ciclo do relógio será determinado pela instrução com tempo de execução maior. Provavelmente a de "load word" que emprega cinco unidades funcionais,**
- **Além de penalidades maiores na execução de instruções de ponto flutuante ou de instruções mais complexas.**
- **Além disto numa implementação mono-ciclo, cada unidade funcional só pode ser usada uma única vez em cada ciclo de relógio.**
- **Uma solução seria duplicar algumas unidades funcionais, mas isto aumentaria o custo de implementação.**

Sugestão de solução

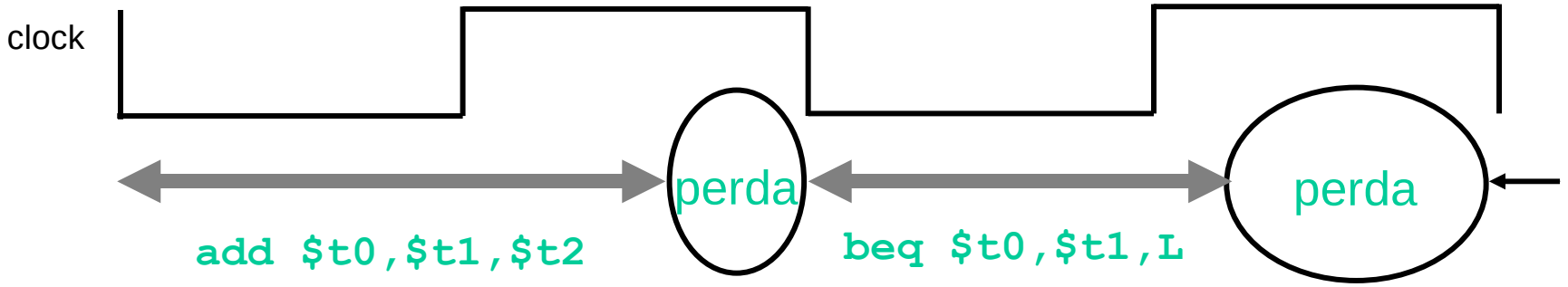
- **Usar técnicas de implementação que tenham um ciclo de relógio mais curto,**
- **Vimos na arquitetura mono-ciclo a execução da instrução nos seus diversos passos,**
- **Podemos usar esses passos para criar uma implementação multiciclo,**
- **Cada passo na execução gastaria 1 período do relógio, ou um ciclo,**
- **Esta execução multiciclo, permite que uma unidade funcional seja utilizada mais de uma vez por instrução, desde que em ciclos diferentes do relógio.**

Vantagens Multiciclo

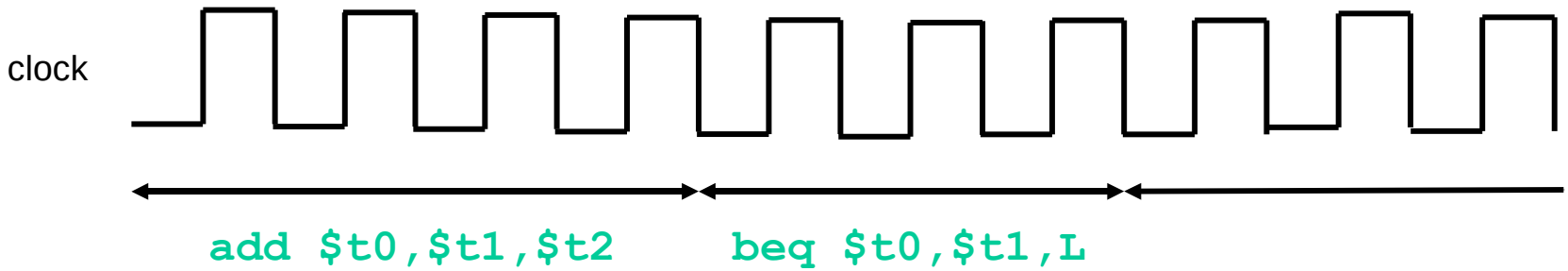
- **Possibilidade de compartilhamento pode ajudar a reduzir o hardware,**
- **Possibilidade de executar uma instrução em ciclos diferentes de períodos do relógio,**
- **Capacidade de compartilhar unidades funcionais no espaço de tempo da execução de uma única instrução**

Relógio: mono-ciclo vs. multiciclo

Implementação mono-ciclo



Implementação Multiciclo



- **Implementação multiciclo: menos perda = maior desempenho**

Bloco operacional – versão multi-ciclo
Caminho de Dados

1. Introdução
2. Ciclos das instruções
3. Bloco operacional completo
4. Execução das instruções
   - Busca de instrução
   - Decodificação de instrução
   - Instruções aritméticas
   - Instruções Load
   - Instruções Store
   - Instruções Branch
   - Instruções Jump
5. Cálculo de desempenho

1. Introdução

- **Máquina mono-ciclo**
  - **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 multi-ciclo**
  - **vários ciclos por instrução**
  - **cada instrução pode ser executada num número diferente de ciclos**
  - **unidades funcionais podem ser reutilizadas em ciclos distintos**
    - **pequeno acréscimo de multiplexadores e registradores**

- **Compromisso no desempenho**
  - **CPI aumenta => desempenho cai**
  - **período do relógio diminui => desempenho sobe**

Introdução

- **Quando é necessário inserir registradores?**
  - **quando valor é computado num 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**
- **Instrução deve ser dividida em passos de duração similar**

2. Ciclos das instruções

1. Busca da instrução

2. Decodificação da instrução

Leitura dos registradores – mesmo que não sejam utilizados

Cálculo do endereço do branch – mesmo que instrução não seja branch

3. Execução da operação – instruções tipo R

Cálculo do endereço efetivo do operando – instruções load e store

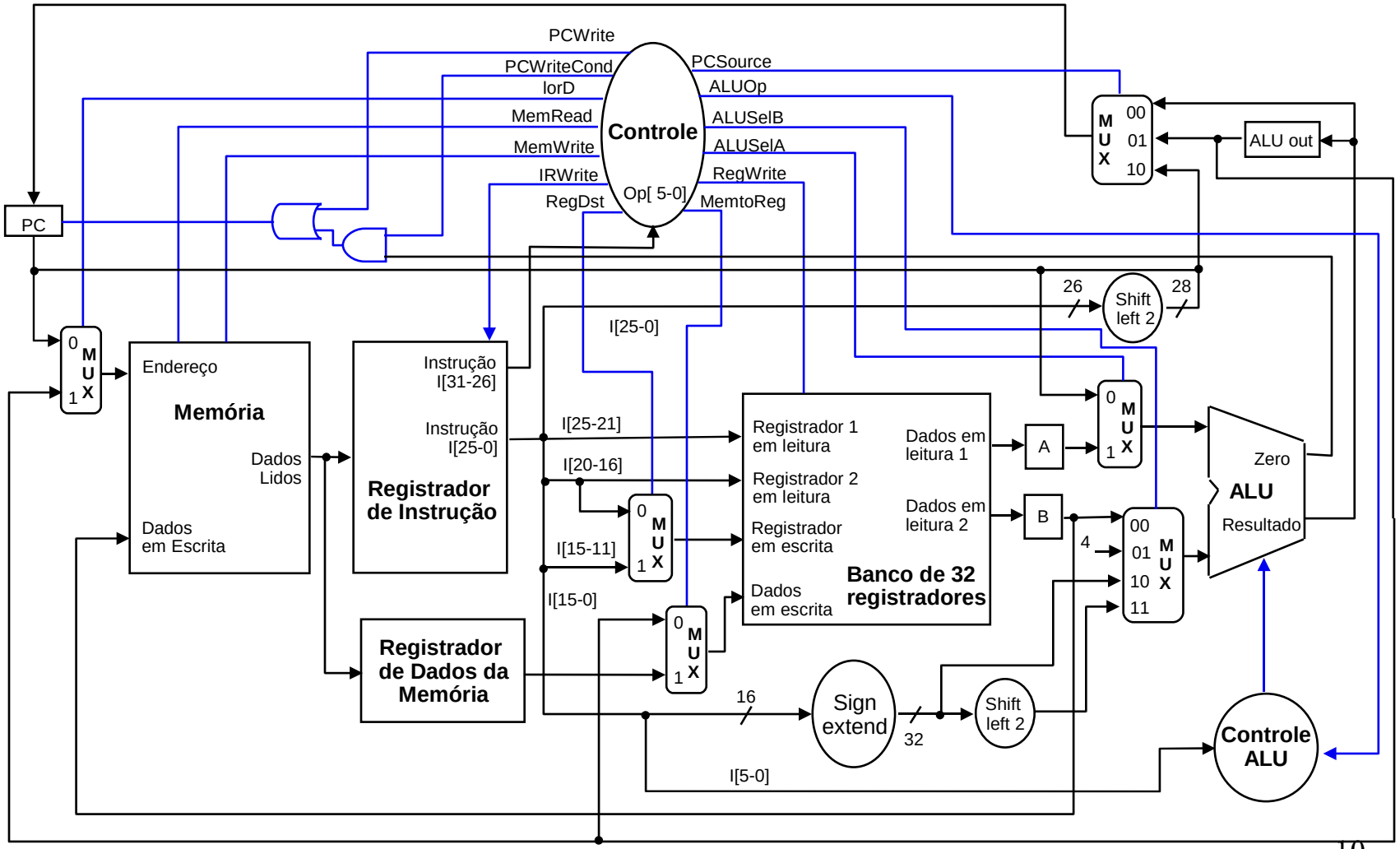
Determinar se branch deve ser executado – instruções branch

4. Acesso à memória – instruções load e store

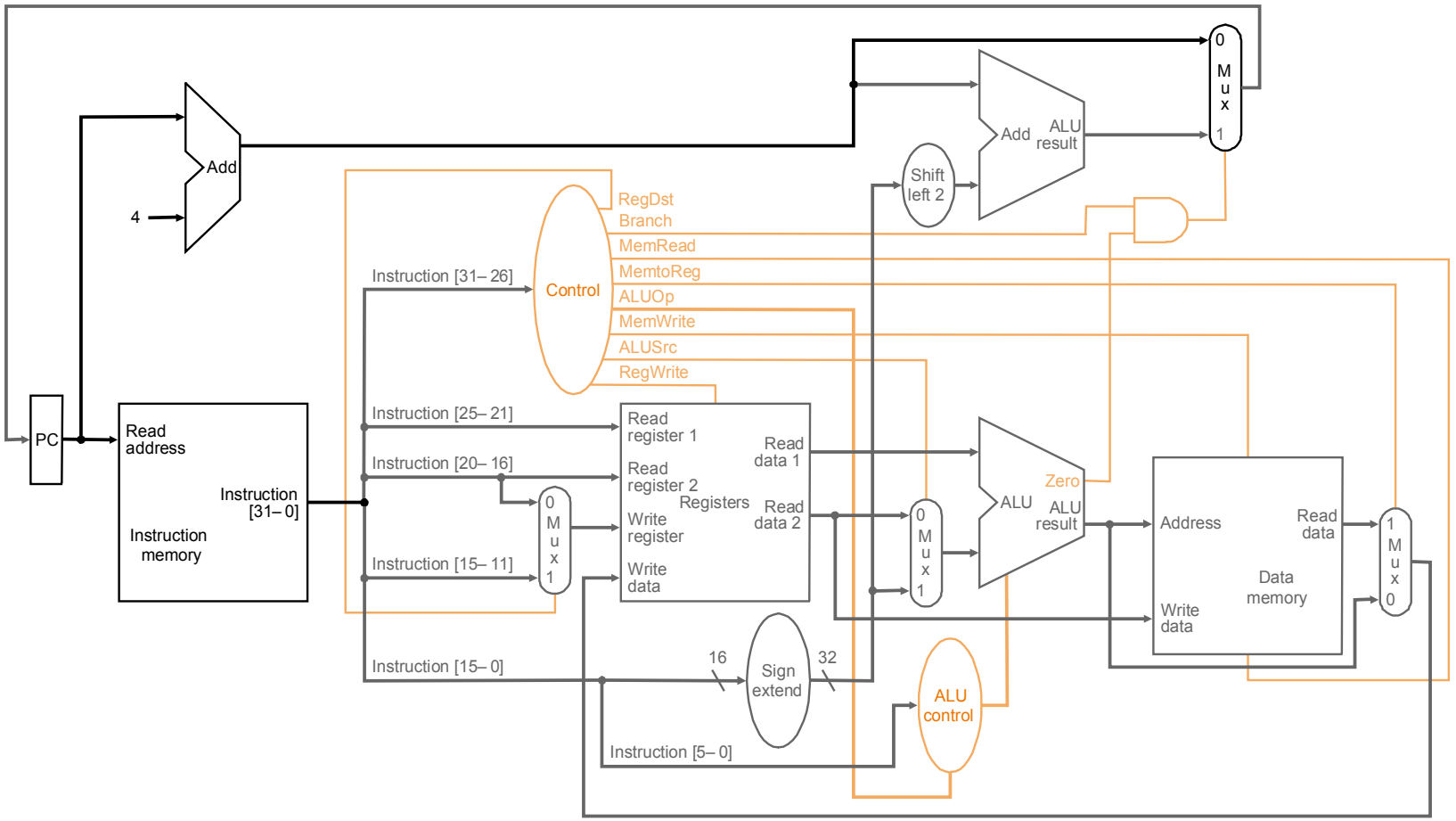
Escrita de registrador – instruções tipo R

5. Escrita de registrador – instruções load

3. Bloco operacional completo Multiciclo



Bloco Operacional Mono-Ciclo



Comparando com BO mono-ciclo

- Uma **única memória** para dados e instruções
- Uma **única ULA unidade lógica e aritmética** para todas as operações
  - operações das instruções tipo R
  - cálculo de 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 ALU
- **Novos multiplexadores** ou extensão dos já existentes

Registradores Adicionados

- **São adicionados ao caminho de dados os seguintes registradores temporários:**
- **O Registrador de Instruções IR,**
- **O Registrador de Dados da Memória MDR,**
- **Os Registradores A e B para guardar os valores dos operandos,**
- **O Registrador UAL Saída para guardar a saída da UAL**

Multiplexadores adicionados

- **Em substituição das 3 UALs da implementação Mono-Ciclo por 1 UAL tornou-se necessário acrescentar:**

- **Um multiplexador na primeira entrada da UAL para escolher entre o registrador A e o PC,**

- **Um multiplexador na segunda entrada da UAL para escolher entre a saída do registrador B que vem do banco de registradores, a constante 4, a extensão do sinal de incremento e a extensão do sinal de incremento deslocado de 2.**

Sinais de controle de 1 bit

Nome do sinal	Efeito quando inativo	Efeito quando ativo
RegDst	O número do registrador-destino no banco de registradores vem do campo rt	O número do registrador-destino no banco de registradores vem do campo rd.
EscReg	Nenhum	O registrador de propósito geral selecionado pelo número do registrador de escrita é atualizado com o valor da entrada Dado de Escrita.
UALFonteA	O primeiro operando da UAL é o PC	O primeiro operando da UAL vem do registrador A.
LerMem	Nenhum	O conteúdo da memória no endereço especificado na entrada Endereço é colocado na saída Dado de Saída.
EscMem	Nenhum	O conteúdo da memória no endereço especificado na entrada Endereço é substituído pelo valor na entrada Dado a ser Escrito.
MemParaReg	O valor colocado na entrada Dado a ser Escrito do banco de registradores vem de UALOut.	O valor na entrada Dado a ser Escrito do banco de registradores vem do MDR.
IouD	O PC é usado para fornecer o endereço da unidade de memória.	UALSaída é usada para fornecer o endereço para a unidade de memória.
IREsc	Nenhum	A saída da unidade de memória é escrita no IR.
PCEsc	Nenhum	O PC é atualizado. A fonte é controlada pelo sinal FontePC.
PCEscCond	Nenhum	O PC é atualizado se a saída Zero da UAL também estiver ativa.

Sinais de Controle de 2 bits

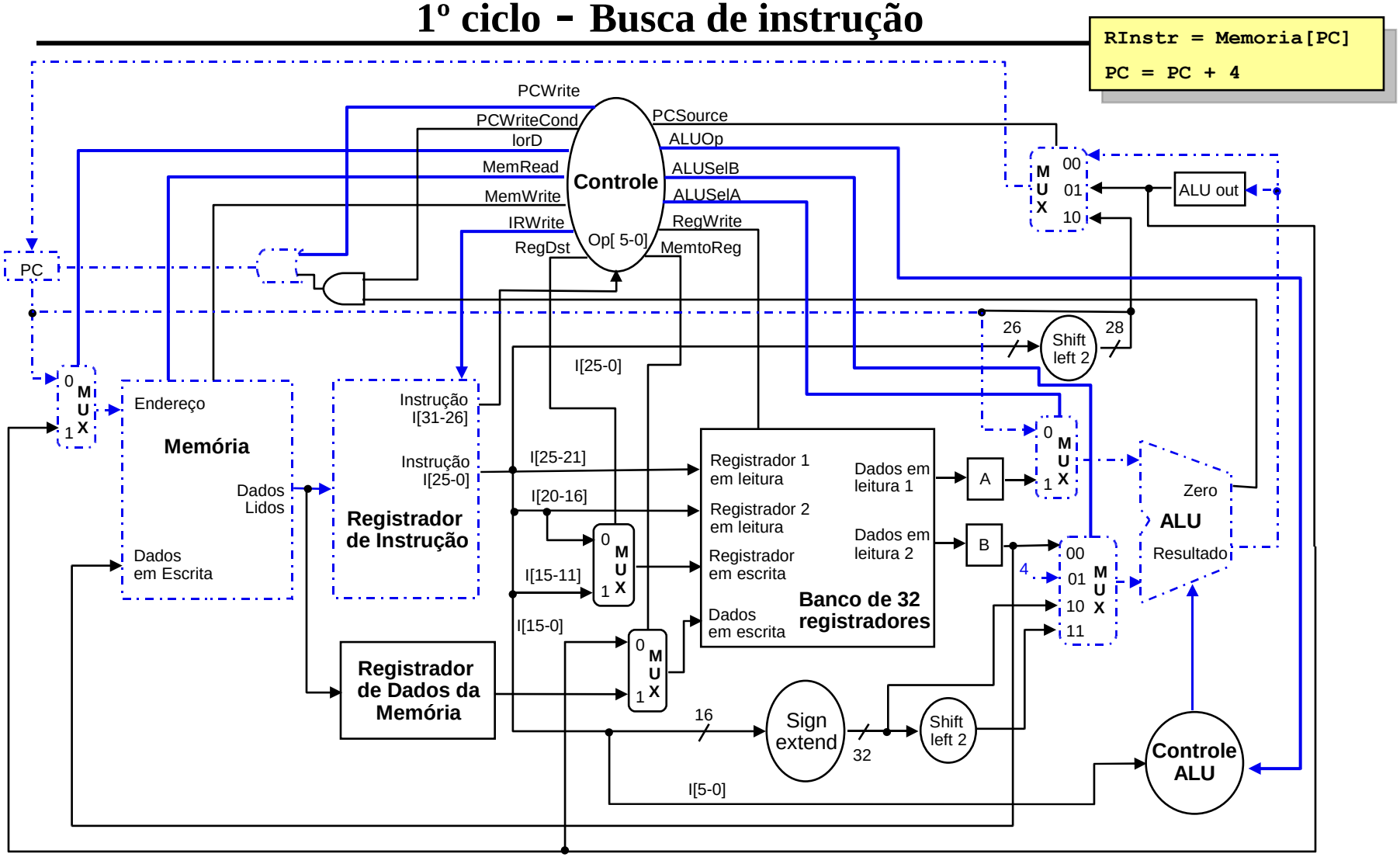
Nome do sinal	Valor	Efeito
UALOp	00	A UAL efetua uma operação de soma.
	01	A UAL efetua uma operação de subtração.
	10	O campo funct (função) da instrução determina a operação da UAL.
UALFonteB	00	A segunda entrada da UAL vem do registrador B.
	01	A segunda entrada da UAL é a constante 4.
	10	A segunda entrada da UAL é a extensão do sinal dos 16 bits menos significativos do IR.
	11	A segunda entrada da UAL é a extensão do sinal dos 16 bits menos significativos do IR deslocados 2 bits à esquerda.
FontePC	00	A saída da UAL (PC + 4) é enviada ao PC para atualizar seu valor.
	01	O conteúdo de UALSaída (o endereço-alvo do desvio condicional) é enviado ao PC para atualizar seu valor.
	10	O endereço-alvo do desvio incondicional (IR[25–0]), deslocado 2 bits à esquerda e concatenado com PC + 4[31–28] é enviado ao PC para atualizar seu valor.

Execução das Instruções

Busca de Instrução

4. Execução das instruções

1º ciclo - Busca de instrução

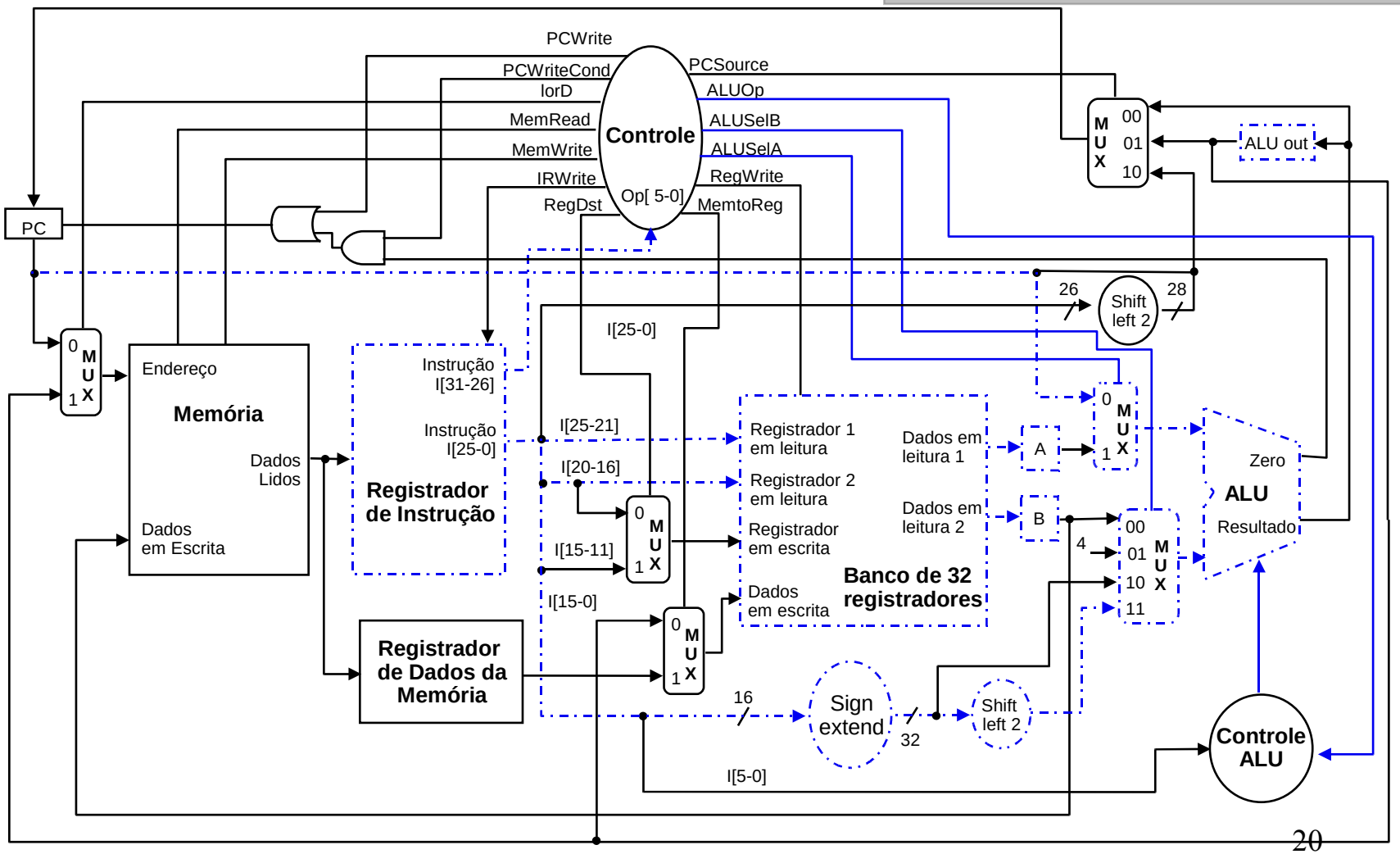


Execução das Instruções

Decodificação de Instrução

2º ciclo - Decodificação de instrução (+ leitura de registradores, + cálculo de end. branch)

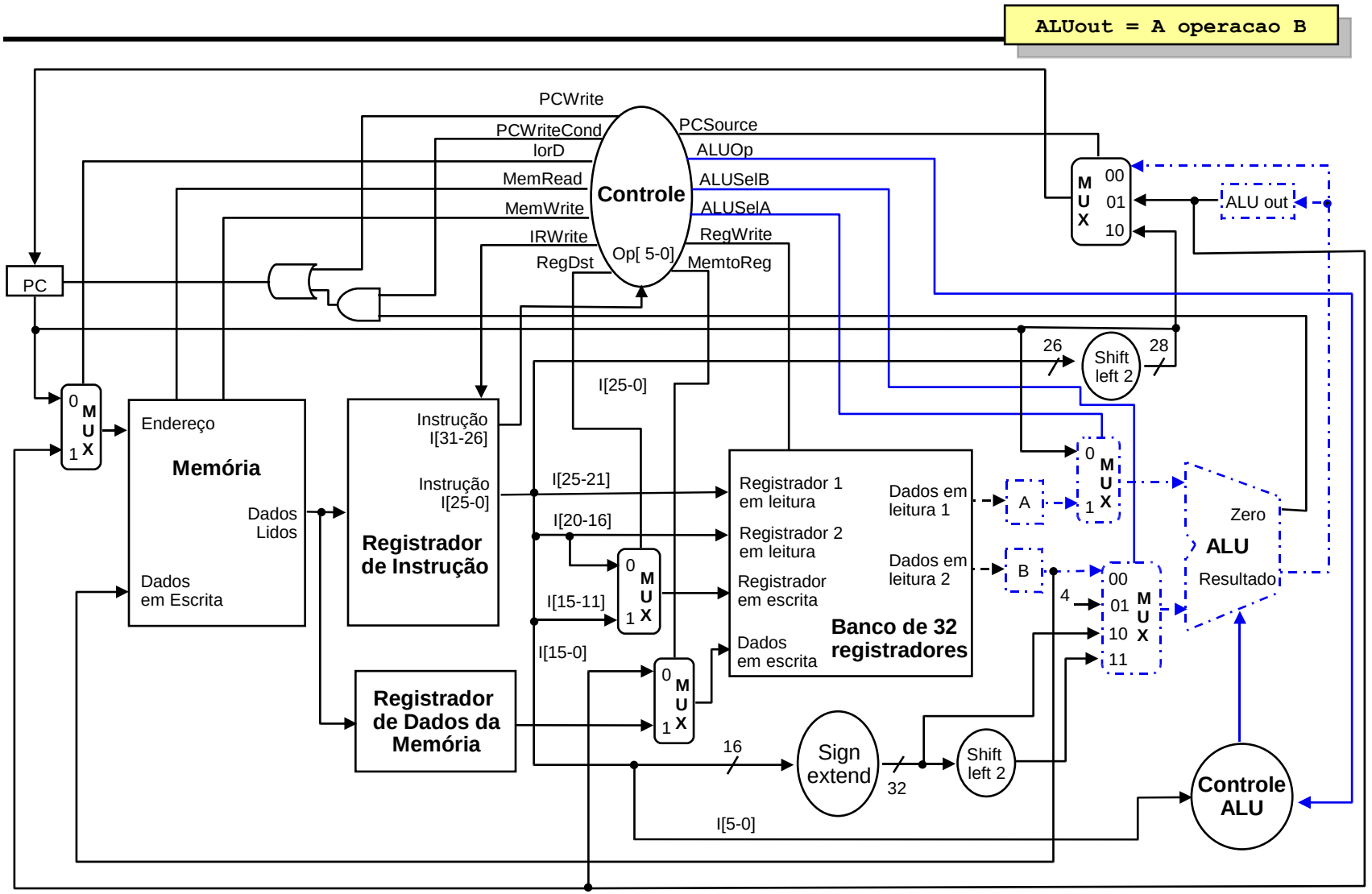
```
A = Reg[I[25-21]]      B = Reg[I[20-16]]
ALUout = PC + SignExtend (I[15-0]) << 2
```



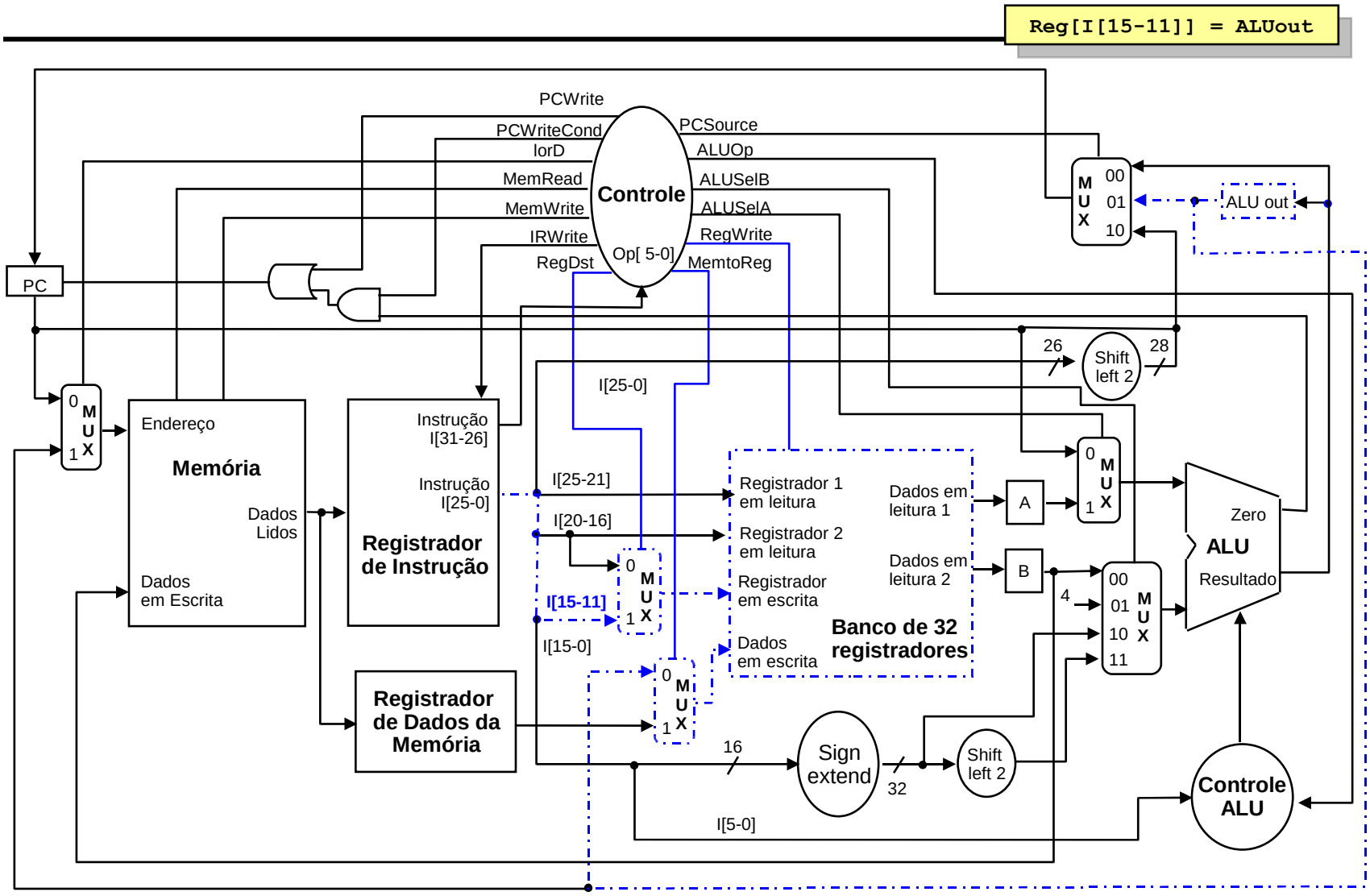
Execução das Instruções

Instruções Aritméticas

Instruções aritméticas – 3º ciclo



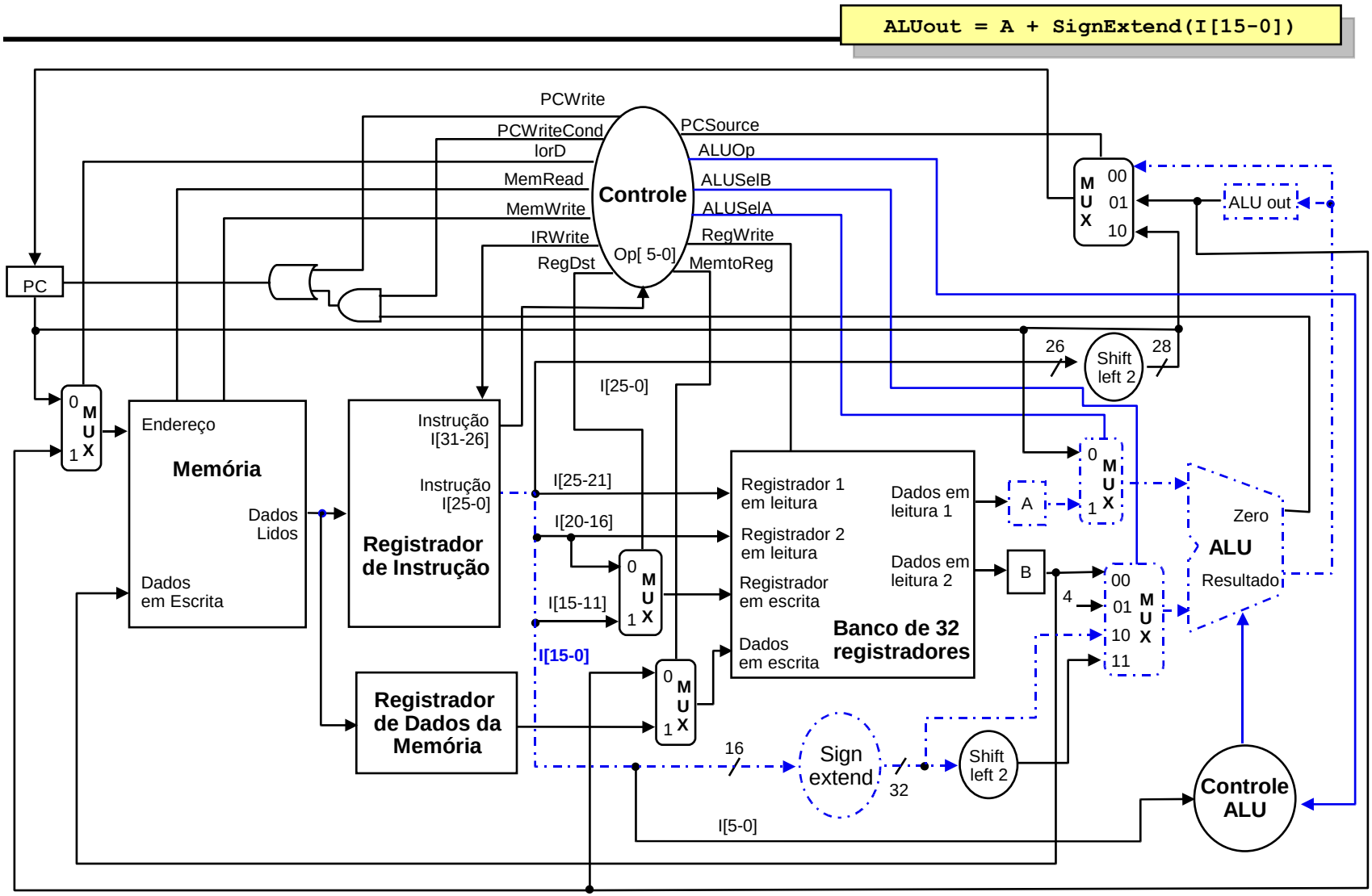
Instruções aritméticas – 4º ciclo



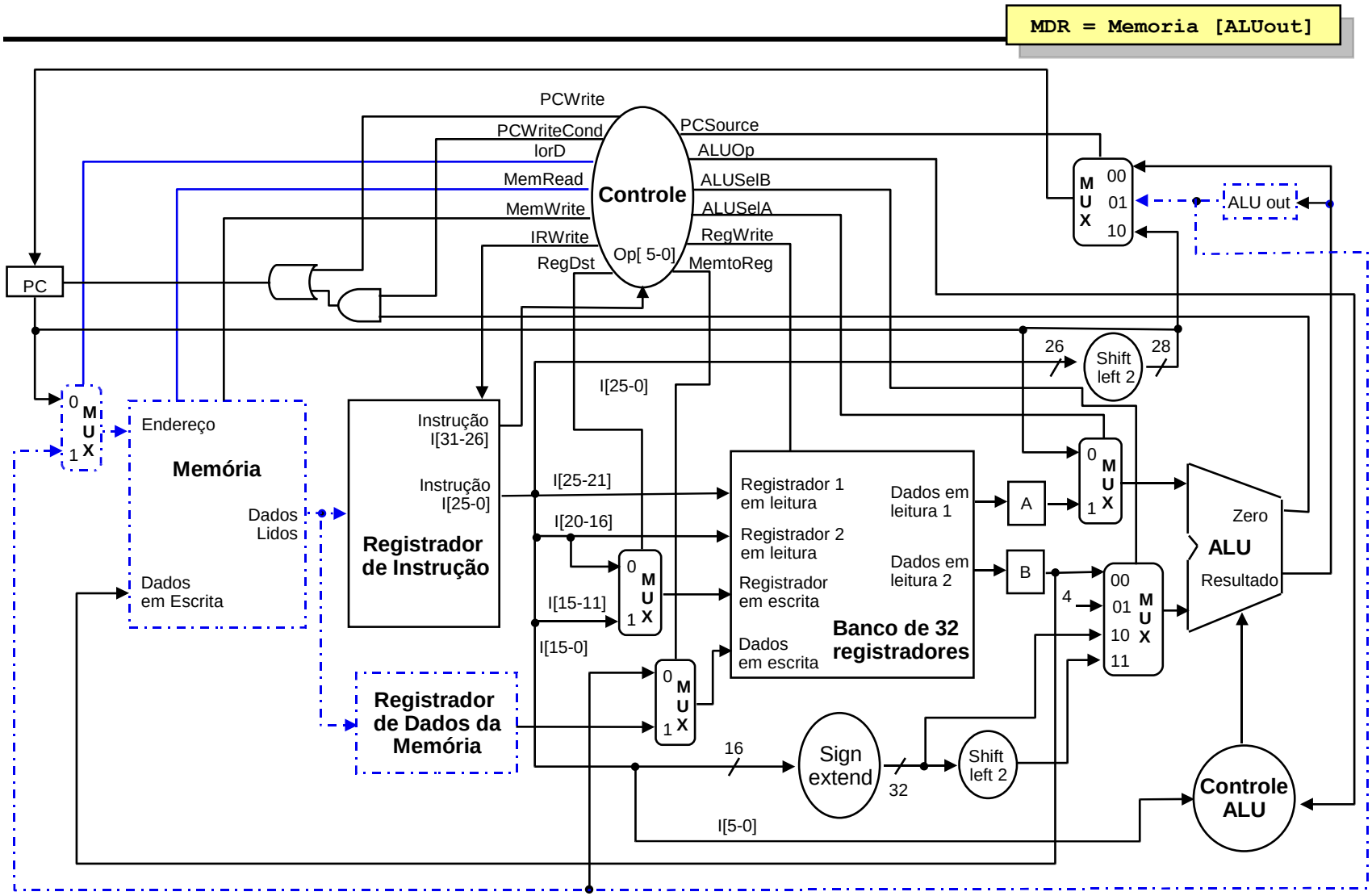
Execução das Instruções

Instruções Load/Store

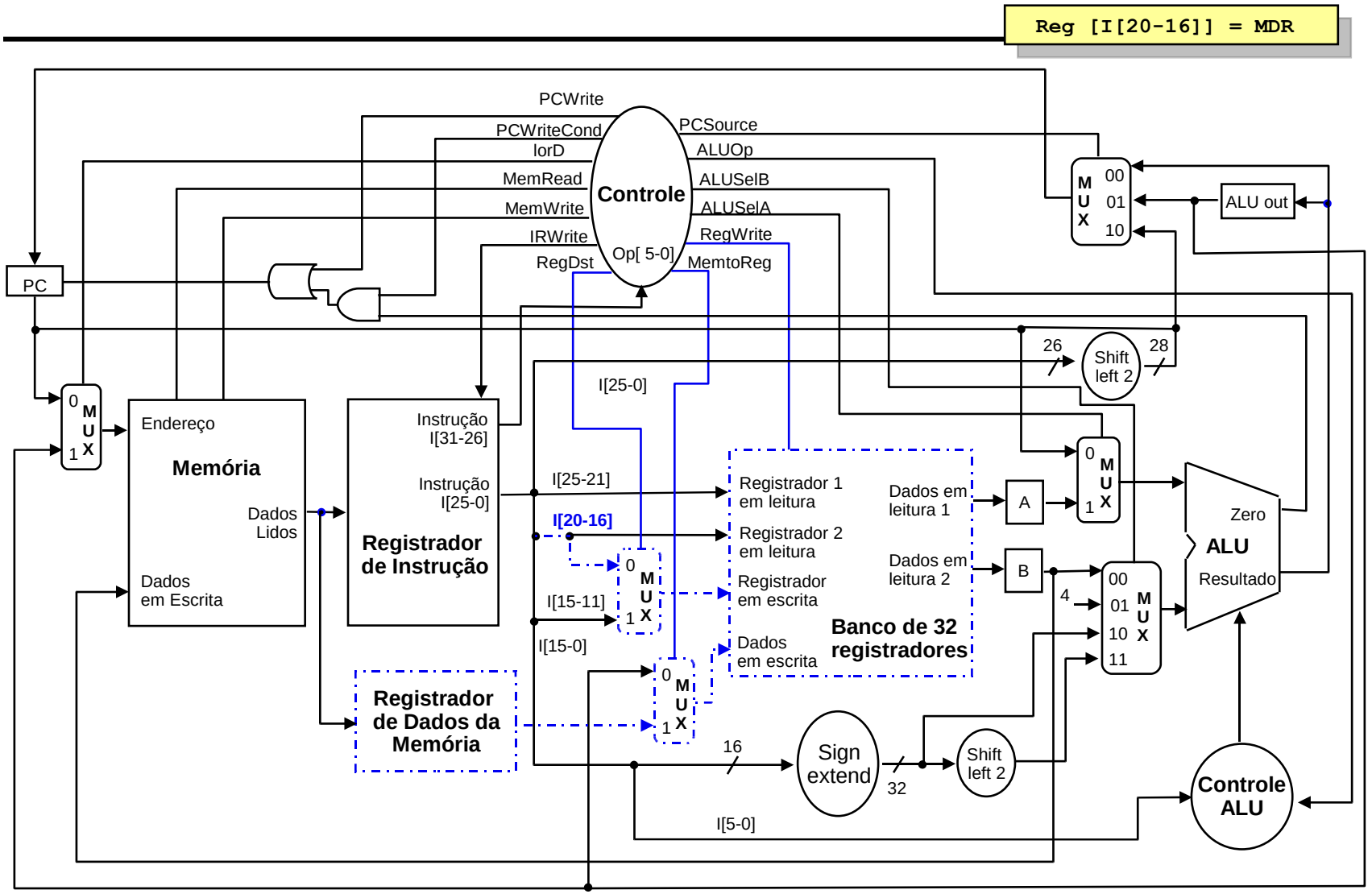
Instruções Load/Store – 3º ciclo



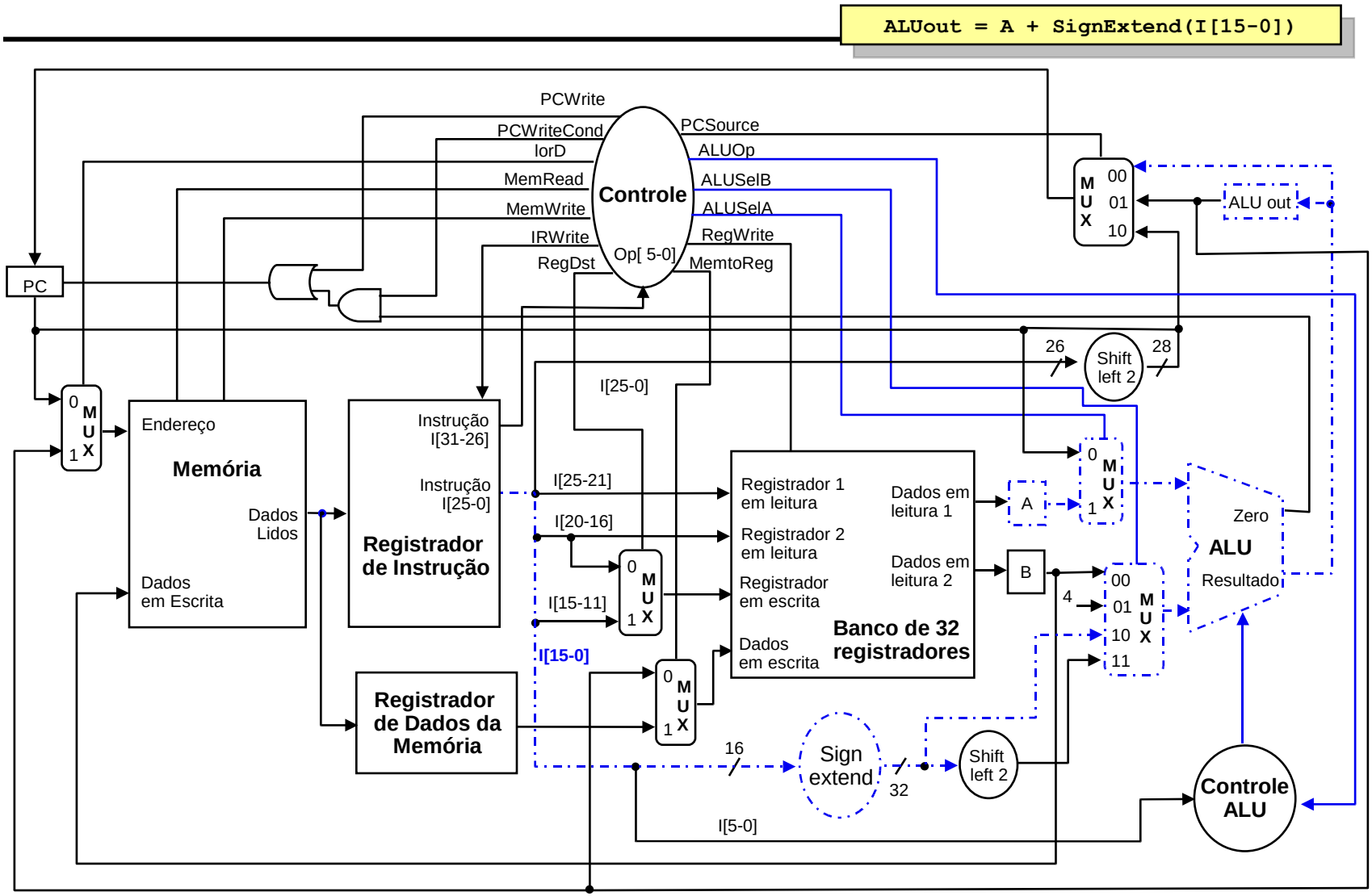
Instrução Load – 4º ciclo



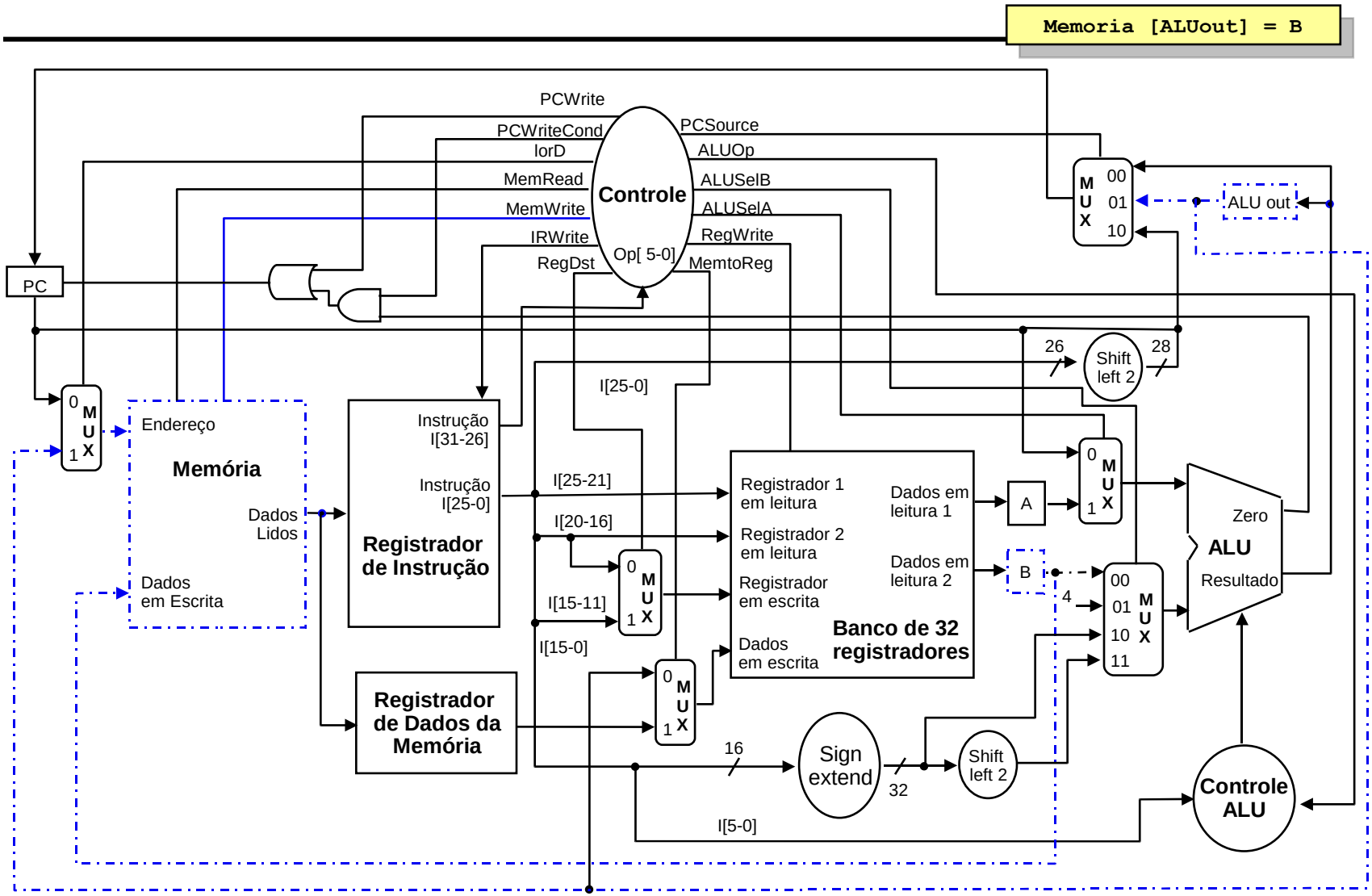
Instrução Load – 5º ciclo



Instruções Load/Store – 3º ciclo



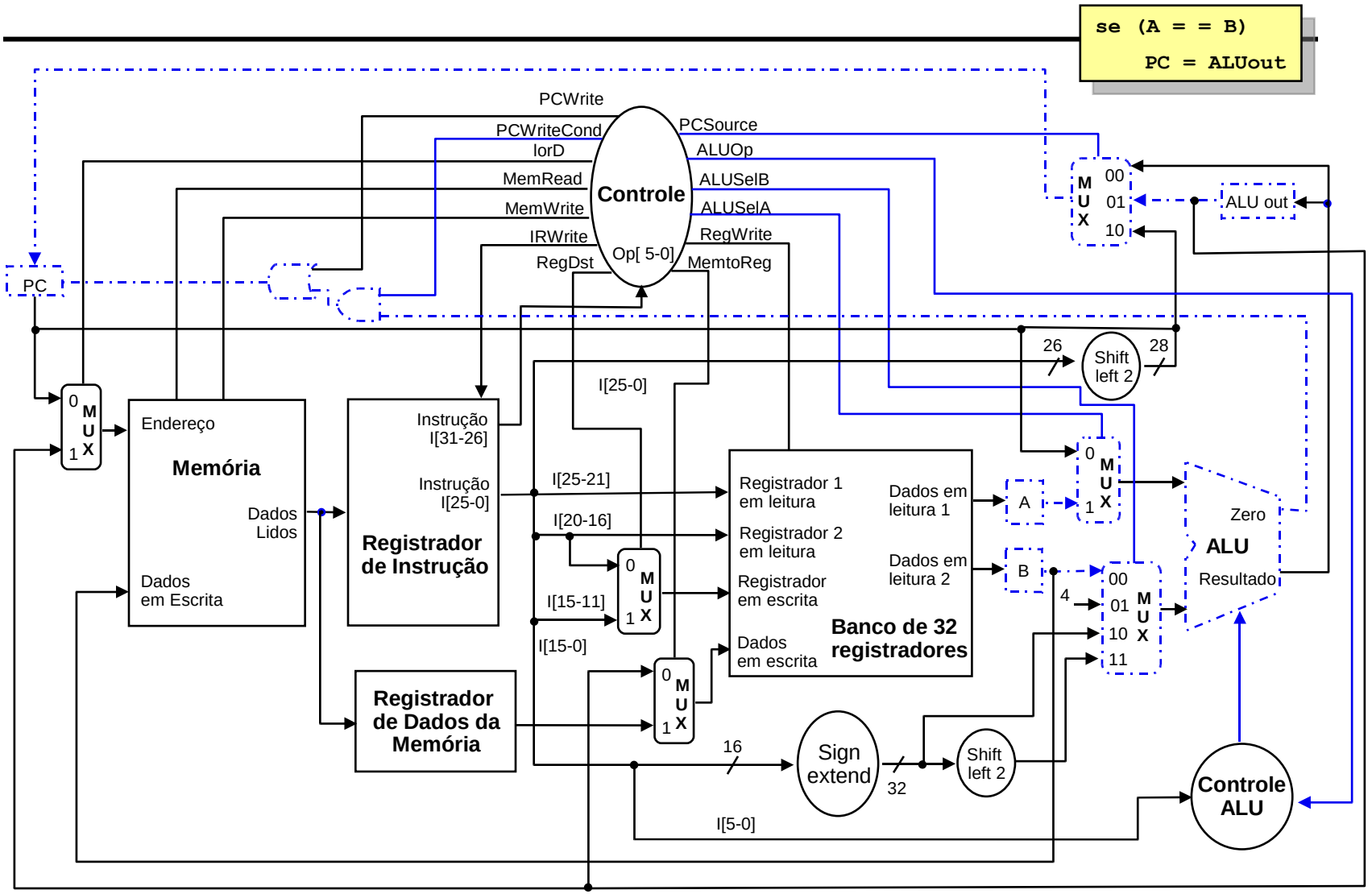
Instrução Store – 4º ciclo



Execução das Instruções

Instrução Branch

Instrução Branch – 3º ciclo

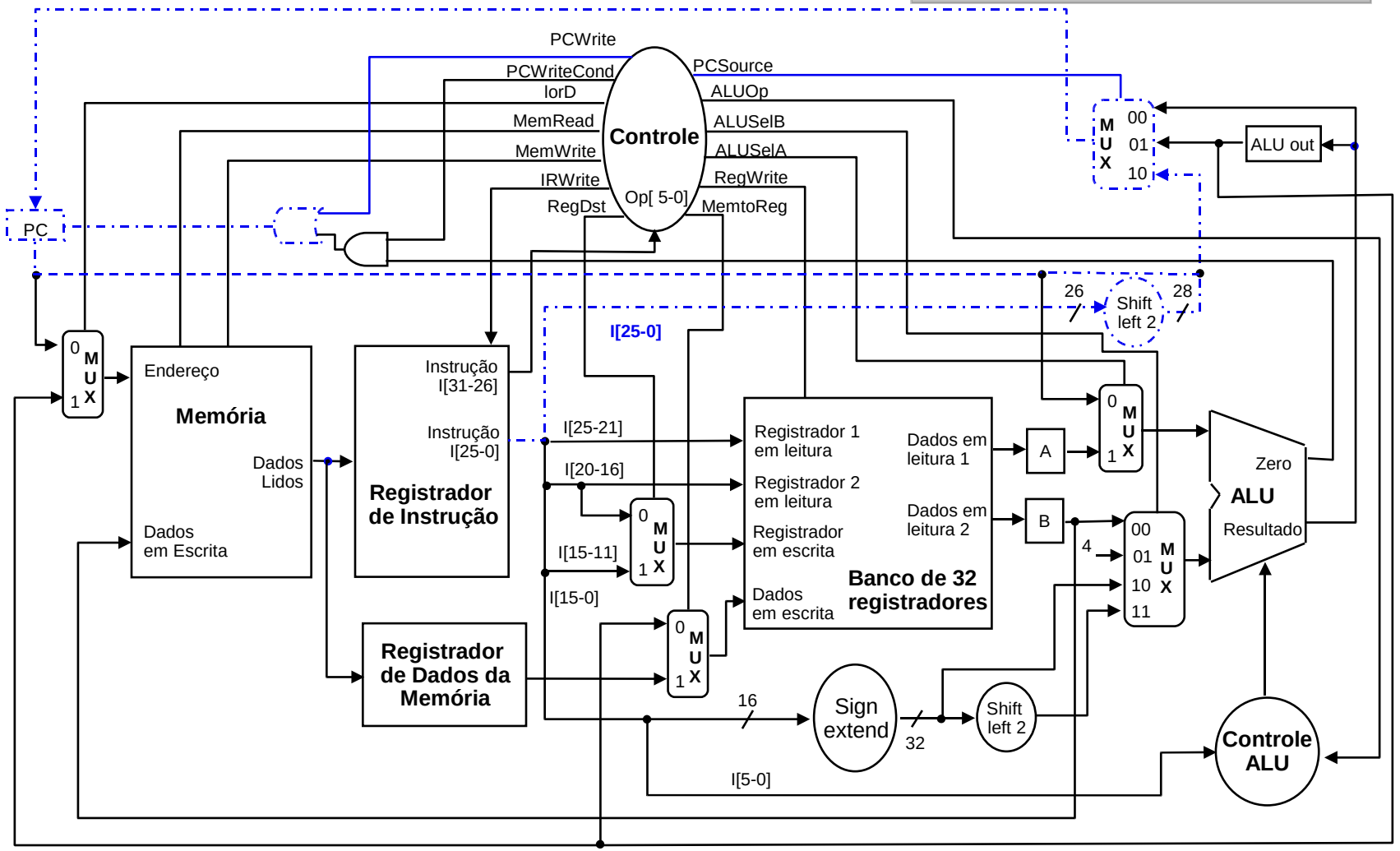


Execução das Instruções

Instrução Jump

Instrução Jump – 3º ciclo

```
PC = PC[31-28] || (I[25-0]<<2)
concatenação
```



Resumo dos passos para executar as instruções

Nome do passo	Ação nas instruções de tipo R	Ação nas instruções de referência à memória	Ação na instrução de desvio condicional	Ação na instrução de desvio incondicional
Busca da instrução	IR = Memória[PC] PC = PC + 4			
Decodificação da instrução/busca dos valores dos registradores	A = Reg [IR[25–21]] B = Reg [IR[20–16]] UALSaída = PC + extensão de sinal (IR[15–0]) << 2)			
Execução, cálculo do endereço, término de uma instrução de branch/jump	ALUOut = A op B	UALSaída = A + extensão de sinal (IR[15–0])	se (A == B) então PC = UALSaída	PC = PC [31–28] II (IR[25–0]<<2)
Término de uma instrução de referência à memória ou de tipo R	Reg [IR[15–11]] = UALSaída	Load: MDR = memória[UALSaída] ou Store: Memória [ALUOut] = B		
Término de leitura da memória		Load: Reg[IR[20–16]] = MDR		

5. Cálculo de desempenho

- **Clock pode ter período de 1 ns (1 GHz), considerando ...**
  - **1 ns para acessos à memória**
  - **0.5 ns para acessos ao banco de registradores**
  - **0.5 ns para operações na ALU**
  - **0 ns para demais blocos (muxs, portas, …)**

- **Com estes valores, período da versão mono-ciclo teria 3.5 ns**
  - **busca da instrução na memória 1ns**
  - **acesso registradores 0,5 ns**
  - **acesso a ULA 0,5 ns**
  - **endereçar memória 1ns**
  - **acesso banco de registradores 0,5 ns**

Cálculo de desempenho

- **O clock da máquina multi-ciclo poderia ser duplicado (2 GHz) se acessos à memória fossem realizados em 2 ciclos de 0.5 ns cada**

- **CPI para cada tipo de instrução**

	versão 1 GHz	versão 2 GHz
– load:	5	7
– store:	4	6
– tipo R:	4	5
– branch:	3	4
– jump:	3	4

Cálculo de desempenho

- **Mix de instruções do compilador gcc**

  22% loads, 11% stores, 49% tipo R, 16% branches, 2% jumps

- **CPI médio na máquina multi-ciclo**

  = 0.22 x 5 + 0.11 x 4 + 0.49 x 4 + 0.18 x 3 = 4.04 na versão 1 GHz

  = 0.22 x 7 + 0.11 x 6 + 0.49 x 5 + 0.18 x 4 = 5.37 na versão 2 GHz

- **Tempo total de execução do programa gcc no MIPS mono-ciclo**

  = N x CPI x período do clock

  = N x 1 x 3.5 ns = 3.5 N x 10^{-9}

- **Tempo total de execução do programa gcc no MIPS multi-ciclo**

  = N x CPI médio x período do clock

  = N x 4.04 x 1 ns = 4.04 N x 10^{-9} na versão 1 GHz

  = N x 5.37 x 0.5 ns = 2.685 N x 10^{-9} na versão 2 GHz

FIM