Capítulo 4

O Processador

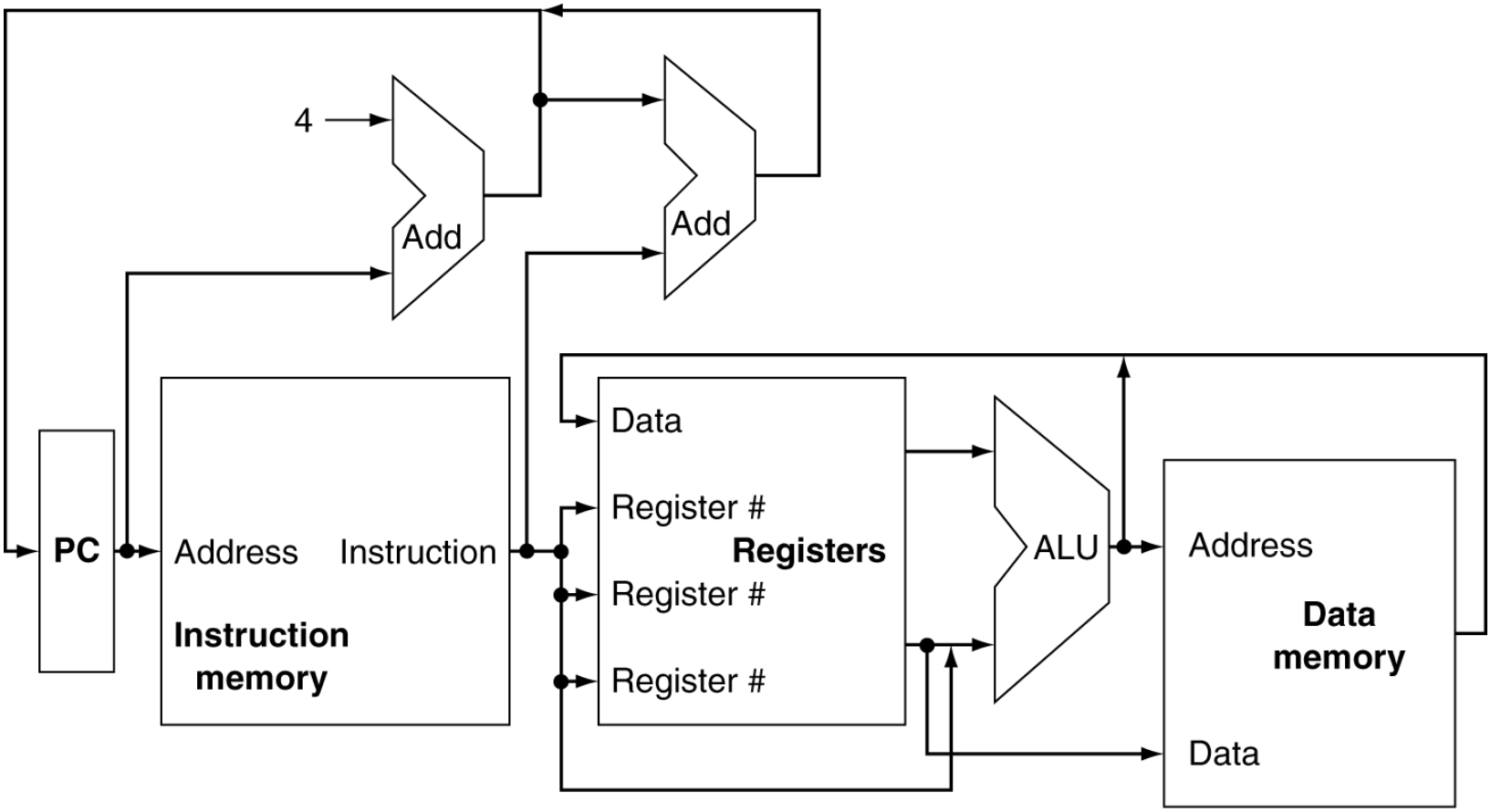
Introdução

- Fatores de desempenho da CPU
  - Contagem de instruções
    - Determinado pelo ISA e pelo compilador
  - CPI e tempo de ciclo
    - Determinado pelo hardware da CPU
- Examinaremos duas implementações do MIPS
  - Uma versão simplificada
  - Uma versão em pipeline mais realista
- Subconjunto simples, mostra a maioria dos aspectos
  - Referência de memória: `lw, sw`
  - Lógica/Aritmética: `add, sub ,and, or, slt`
  - Transferência de controle: `beq, j`

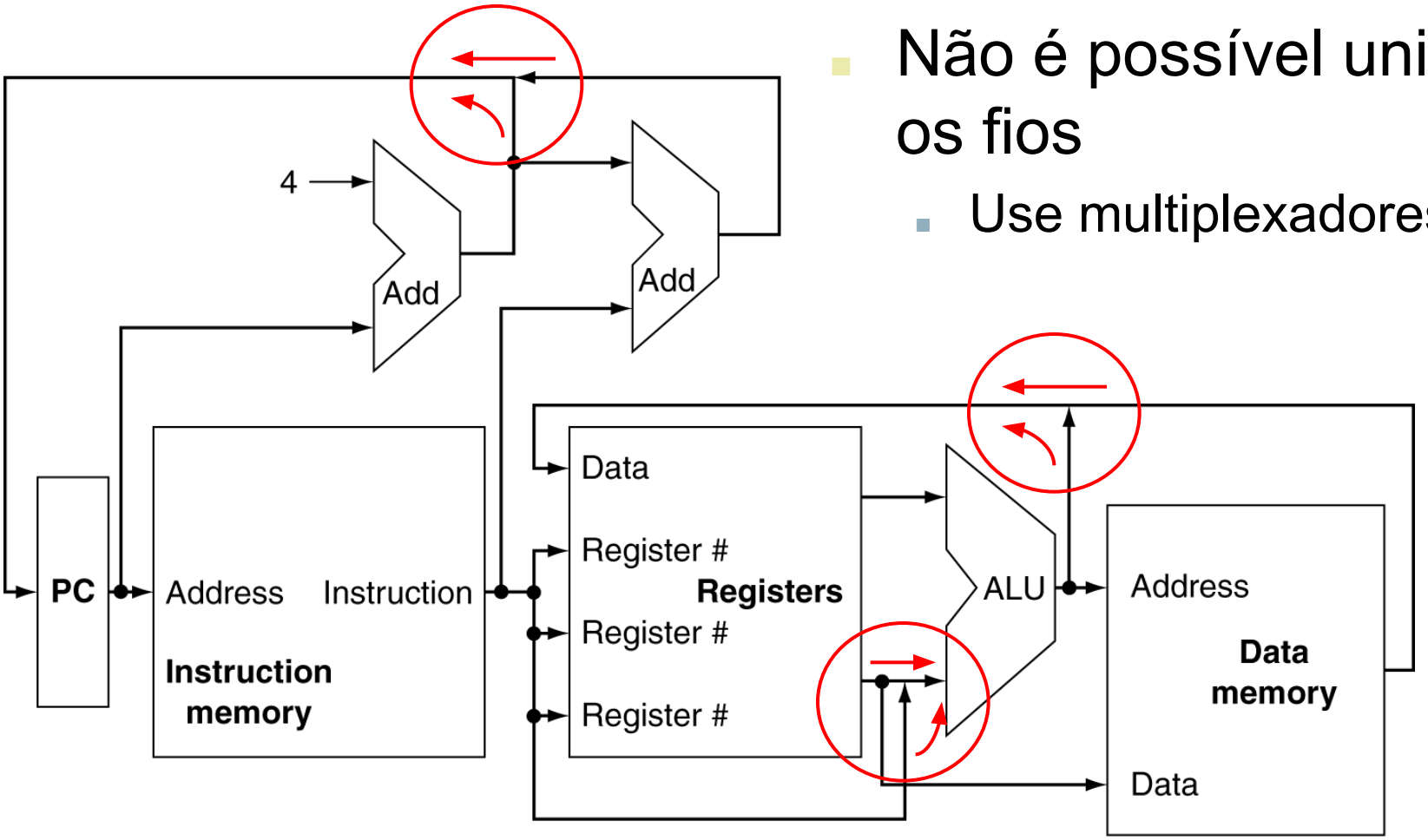
Execução de Instrução

1. PC → endereço da instrução a ser buscada na memória. Atualizar PC para próxima instrução
2. Leitura dos registradores
3. Dependendo da classe de instrução usar a ULA para calcular:
   a. Resultado aritmético
   b. Endereço de memória para load/store
   c. Endereço de desvio condicional (branch)
4. Acessar memória de dados para load/store
5. Escrever resultado no registrador destino

CPU Overview

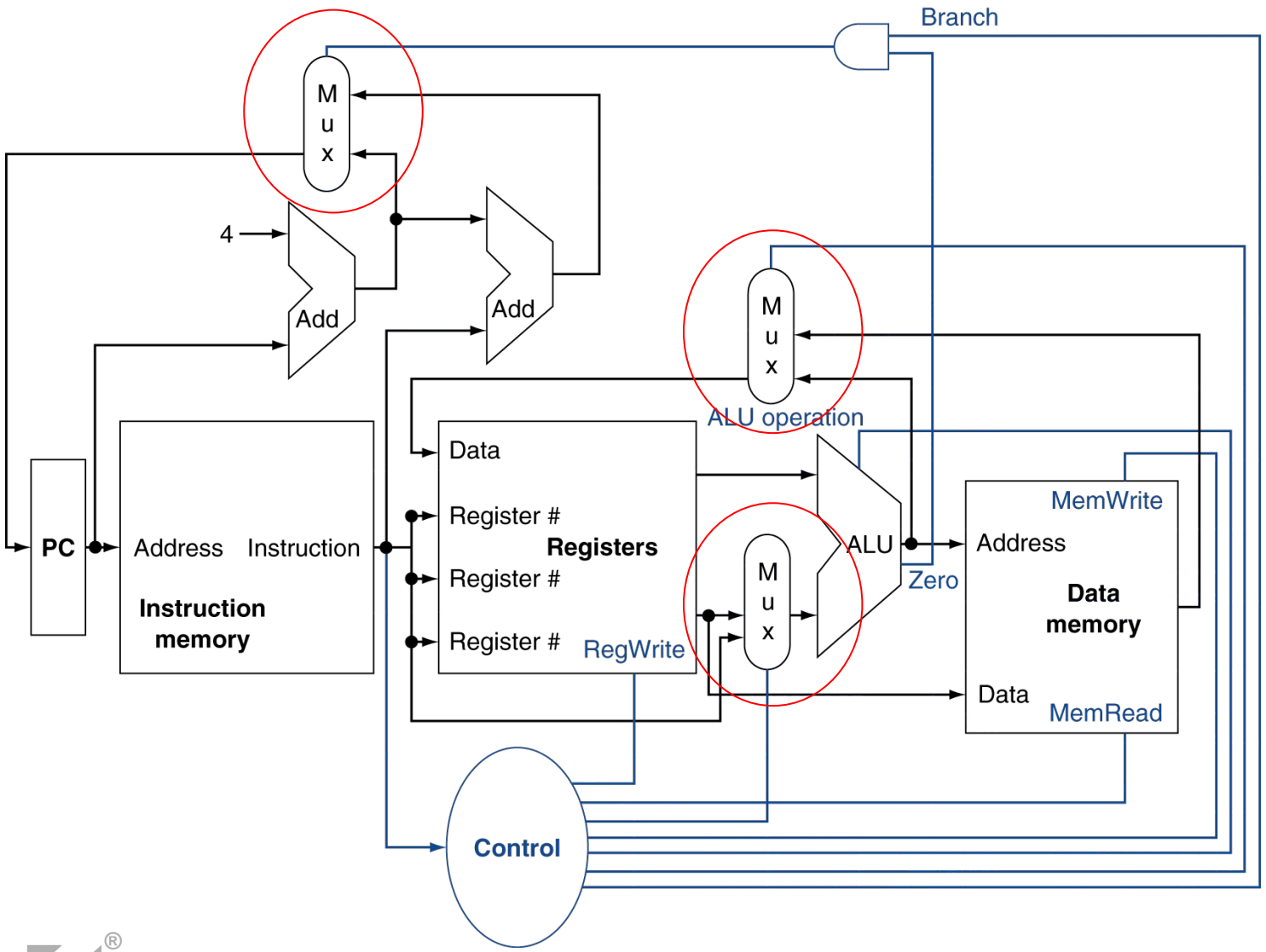


Multiplexadores



- Não é possível unir os fios
  - Use multiplexadores

Multiplexadores

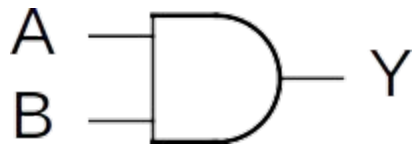


Noções básicas de design de lógica

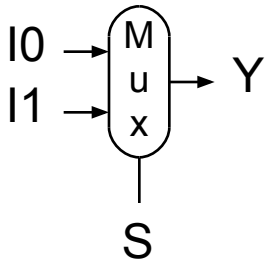
- Informações codificadas em binário
  - Baixa tensão = 0, Alta tensão = 1
  - Um fio por bit
  - Dados de vários bits codificados em barramentos de vários fios
- Elemento combinacional
  - Opera com dados
  - Saída é uma função da entrada
- Elementos de estado (sequenciais)
  - Guarda informação

Elementos Combinacionais

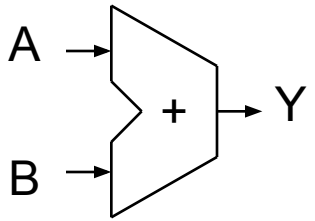
- Porta AND
  - Y = A & B



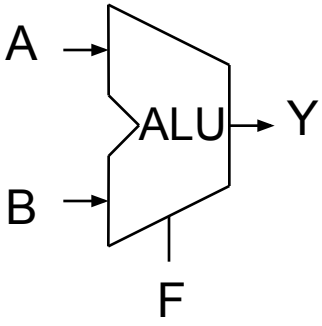
- Multiplexador
  - Y = S ? I1 : I0



- Somador
  - Y = A + B

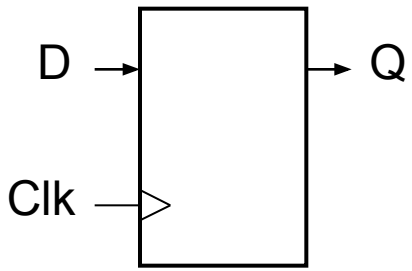


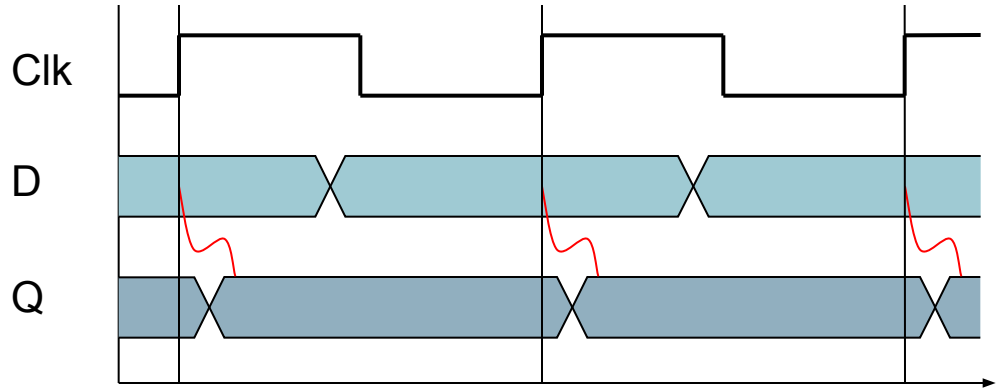
- Unidade Lógica/Aritmética
  - Y = F(A, B)



Elementos Sequenciais

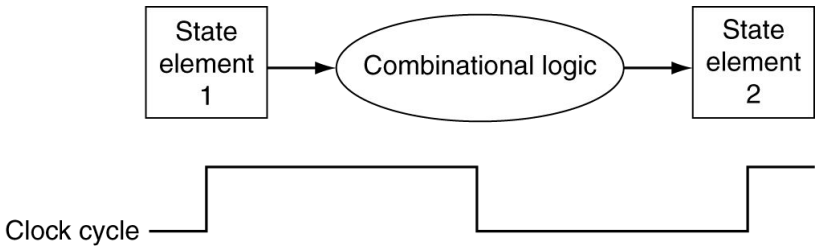
- Registrador: armazena dados em um circuito
  - Usa um sinal de relógio para determinar quando atualizar o valor armazenado
  - Acionado por borda: atualização quando Clk muda de 0 para 1

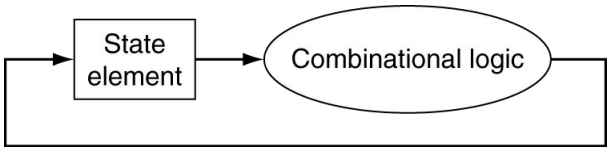




Funcionamento do Clock

- A lógica combinacional transforma dados durante ciclos de clock
  - Entre as bordas do relógio
  - Entrada de elementos de estado, saída para elemento de estado
  - O atraso mais longo determina o período do relógio





Construindo um Datapath

- Datapath
  - Elementos que processam dados e endereços na CPU
    - Registradores, ULAs, mux's, memórias,…
- Vamos construir um caminho de dados MIPS incrementalmente
  - Refinando o design da visão geral

Datapath Completo

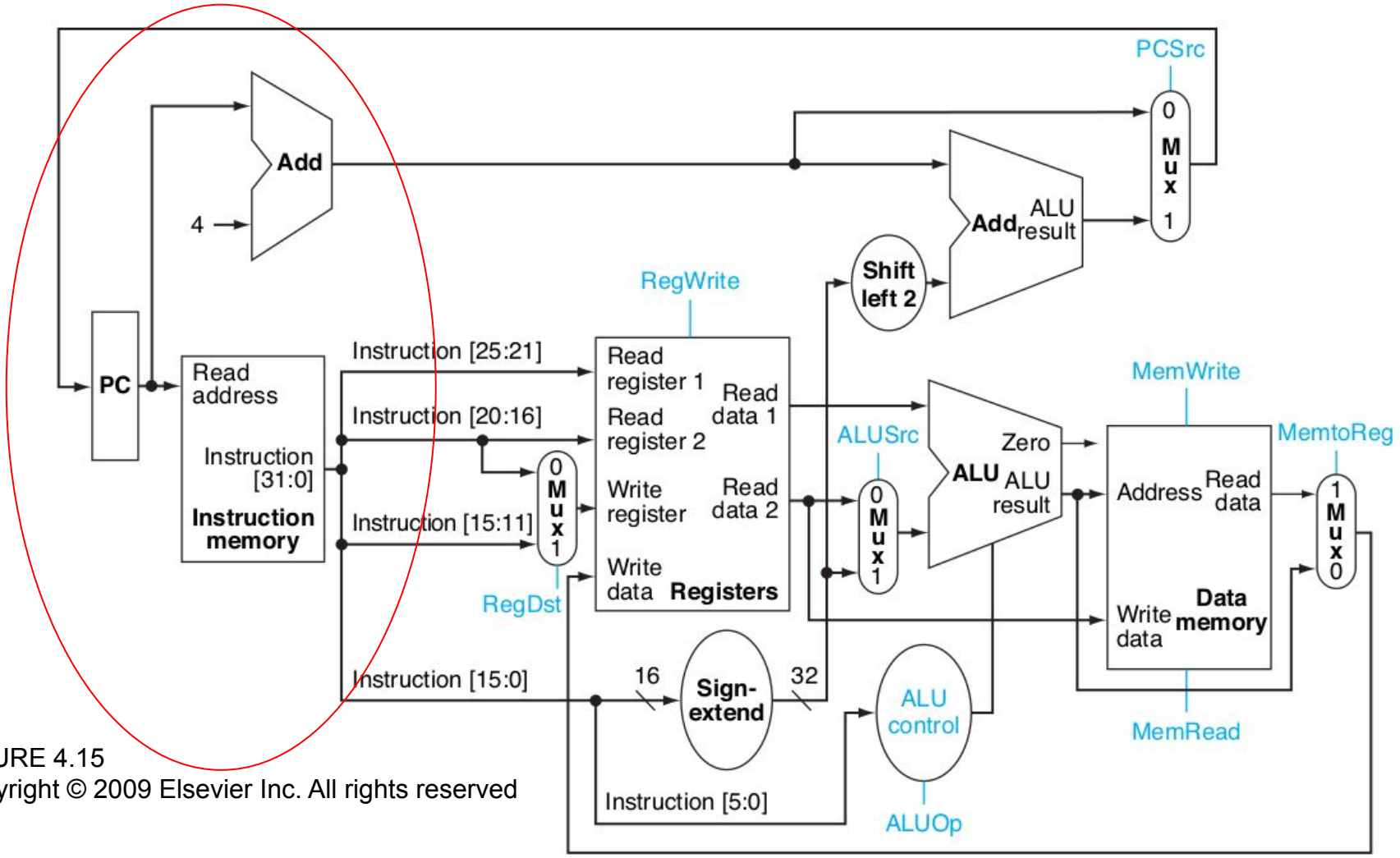
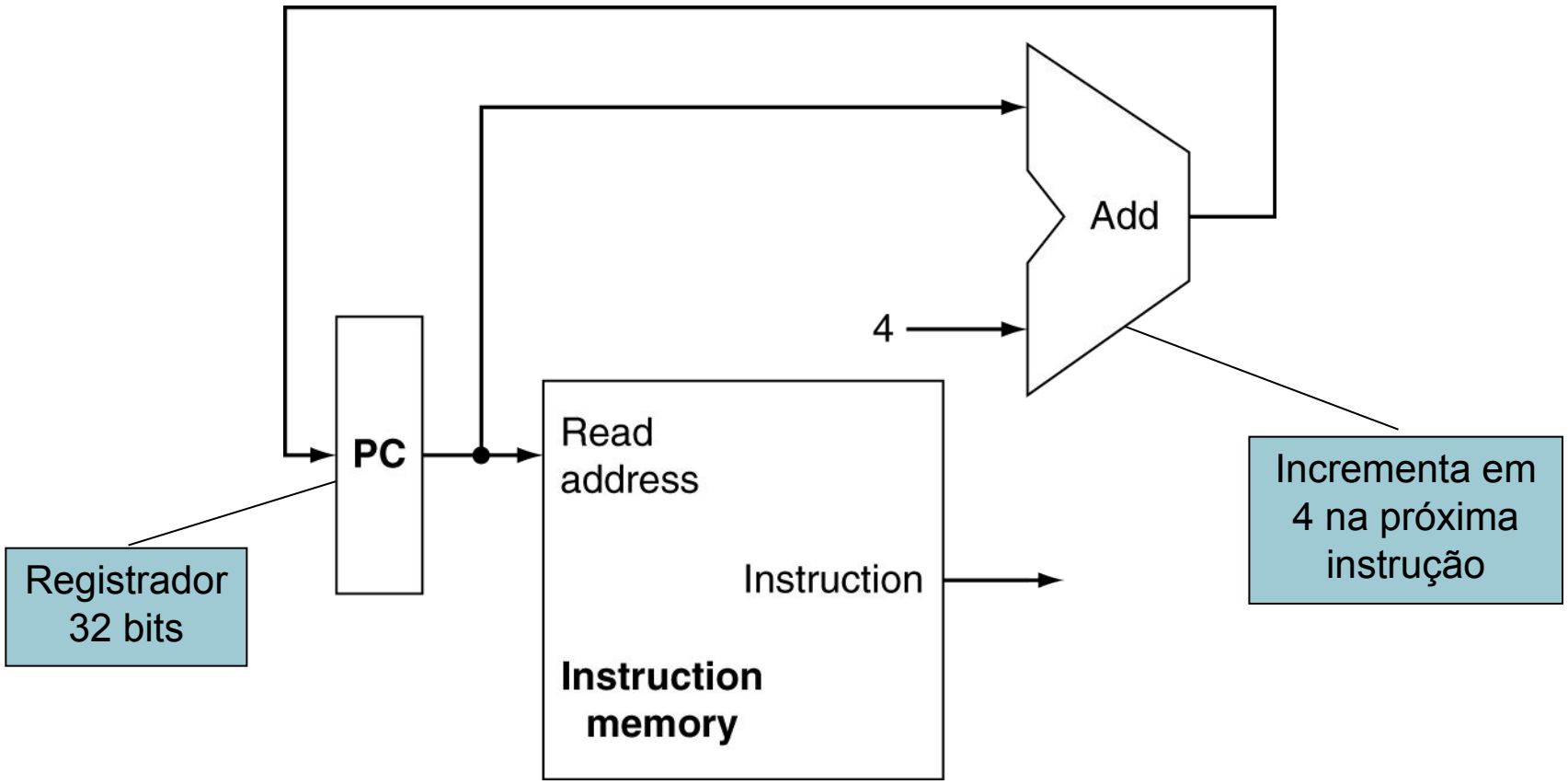
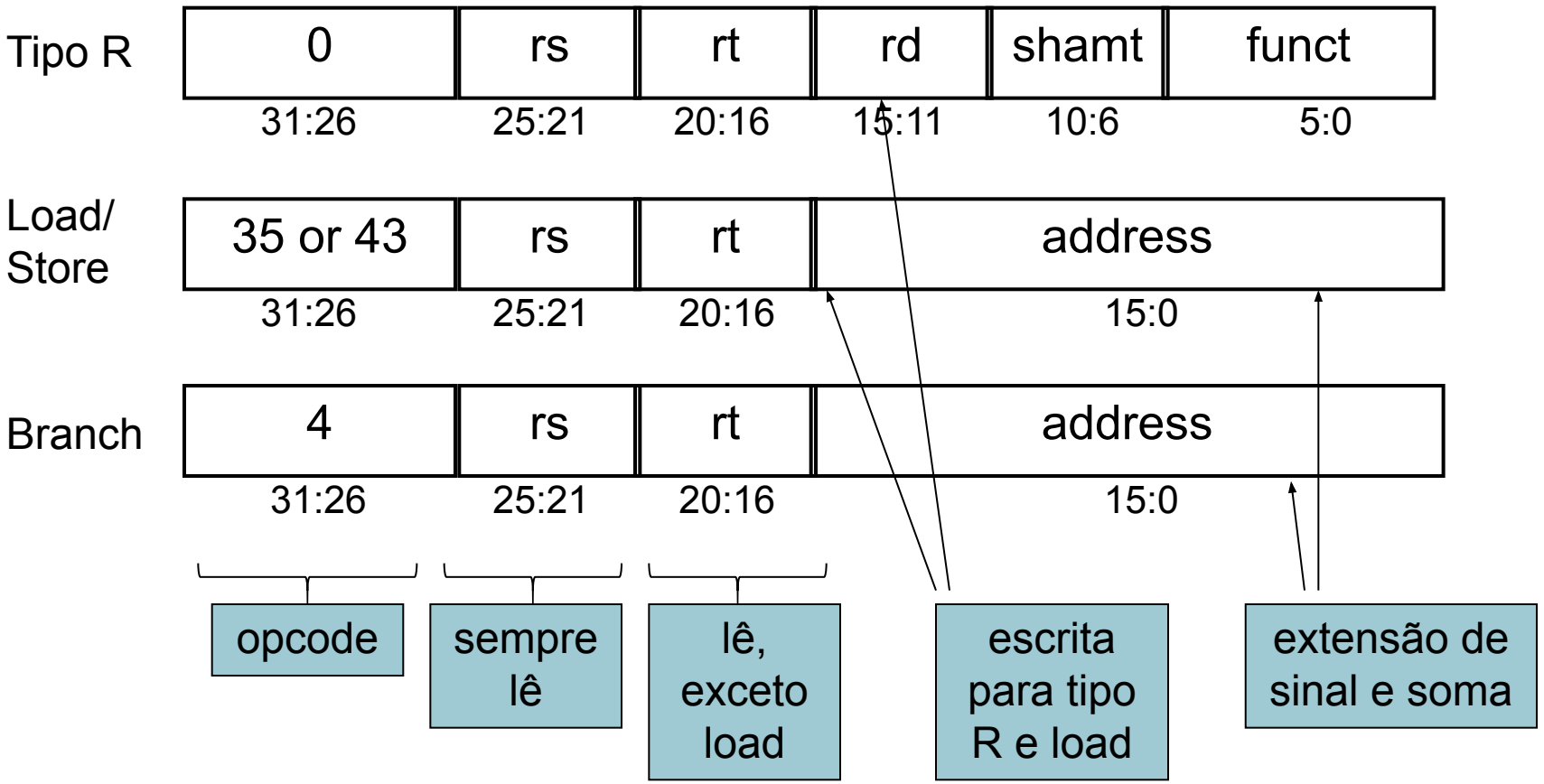


FIGURE 4.15

1- Busca de Instruções

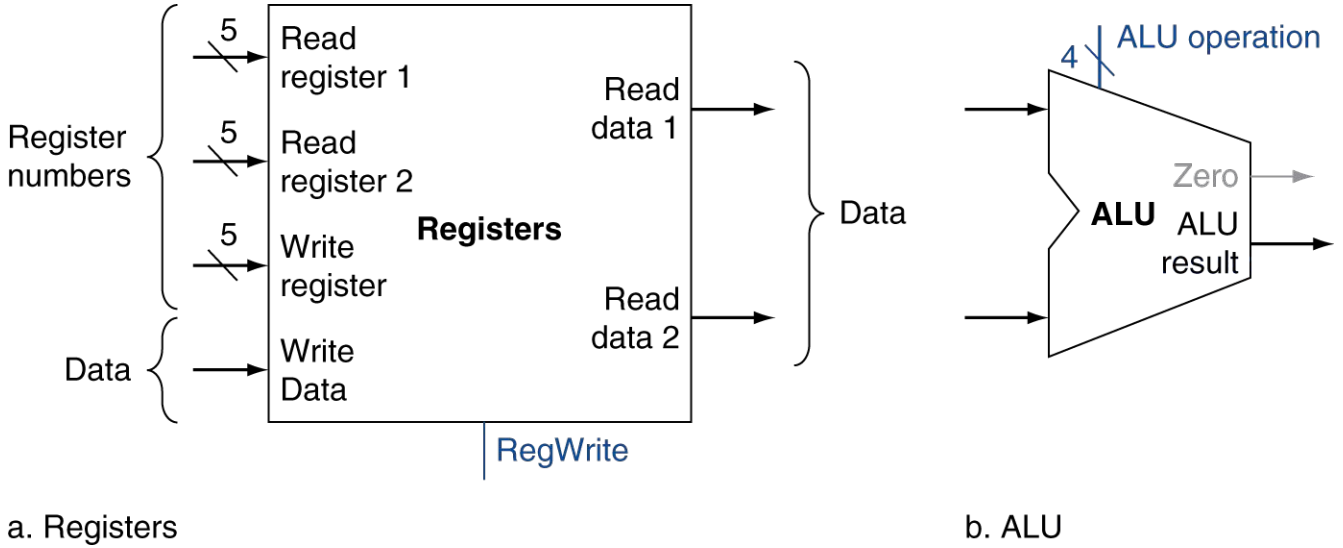


2 - Leitura de Operandos



Instruções Formato R

- Lê dois operandos dos registradores
- Executa operação lógica/aritmética
- Escreve o resultado no registrador



Instruções Formato R

0x00400014 add $t1,$t1,$s6 (add $9,$9,$22)																R[rd] = R[rs] + R[rt] (0/20 hex)											
0x00						9					22					9										(0x20)	
0	0	0	0	0	0	0	1	0	0	1	1	0	1	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0
0				1				3				6				4				8				2				0	

Instruções Formato R - Busca

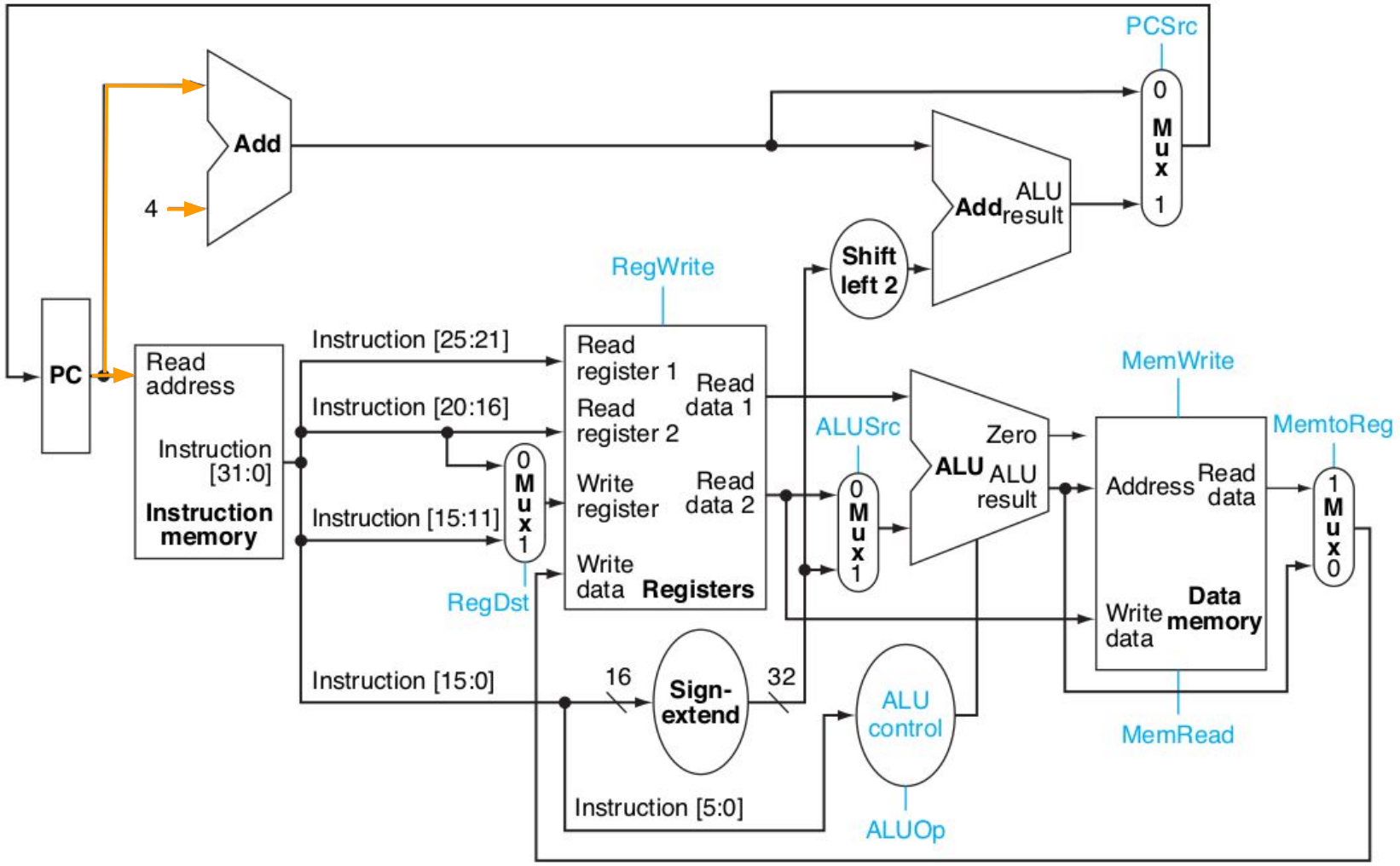


FIGURE 4.15

Instruções Formato R - Busca

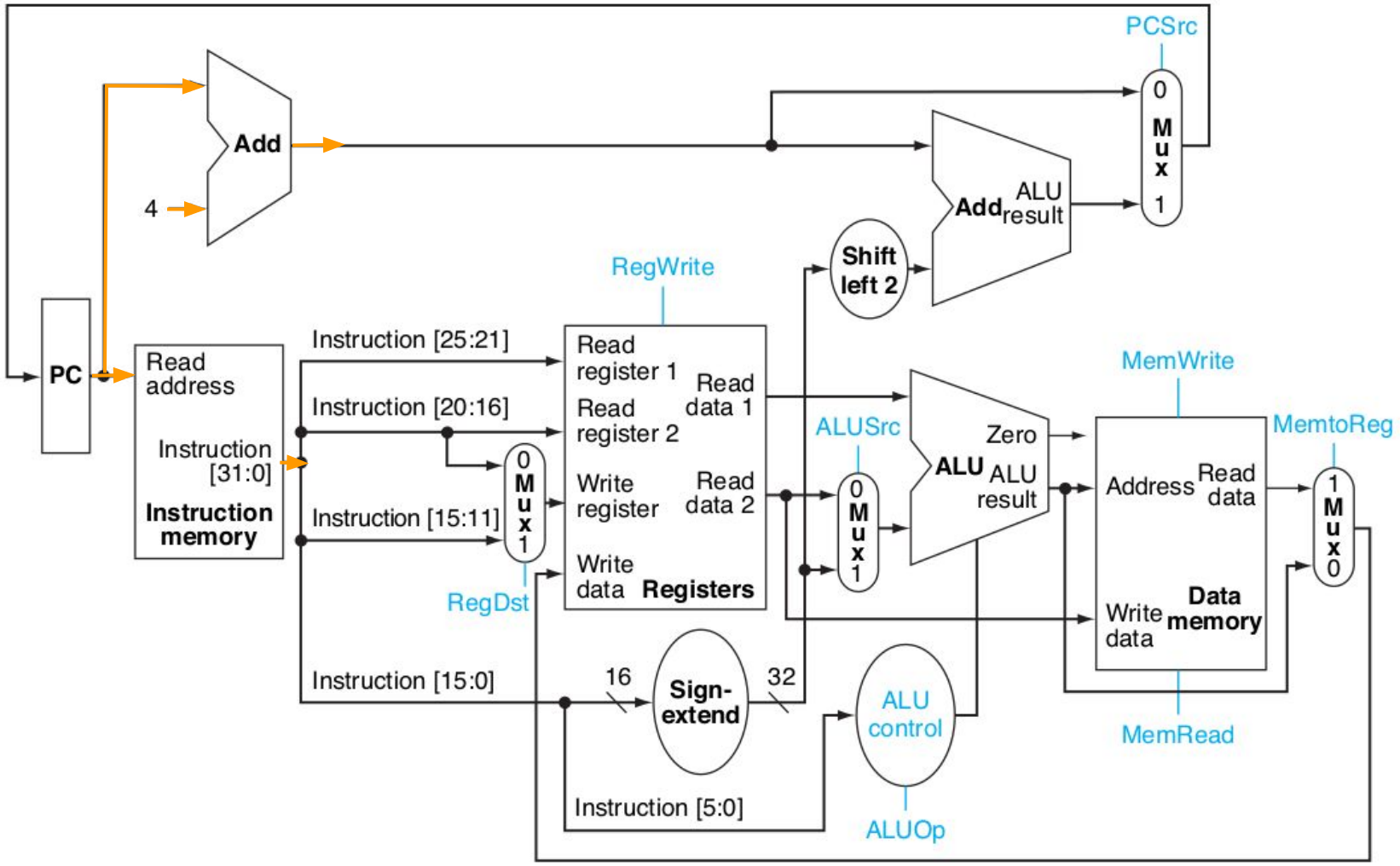
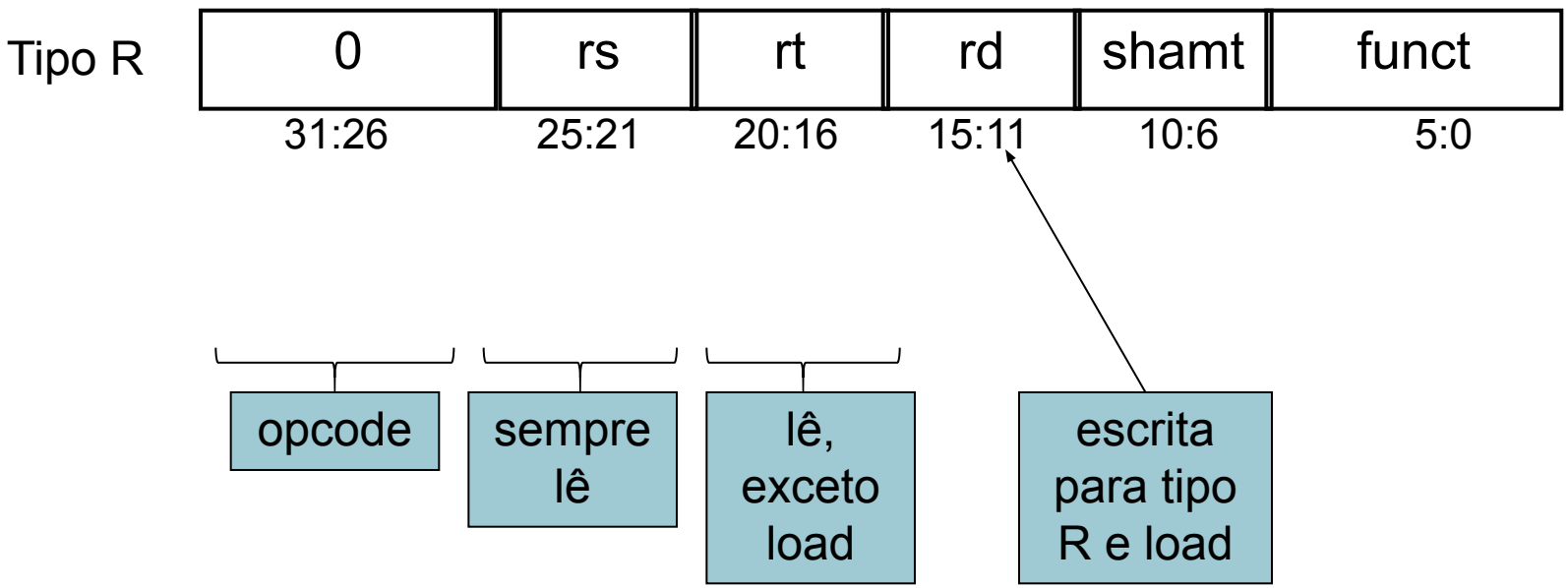


FIGURE 4.15

2 - Leitura de Operandos

0x00400014 add $t1,$t1,$s6 (add $9,$9,$22)																		R[rd] = R[rs] + R[rt] (0/20 hex)											
0x00						9					22					9										(0x20)				
0	0	0	0	0	0	0	1	0	0	1	1	0	1	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0
0					1				3				6				4				8				2				0	



Instruções Formato R - Leitura Reg

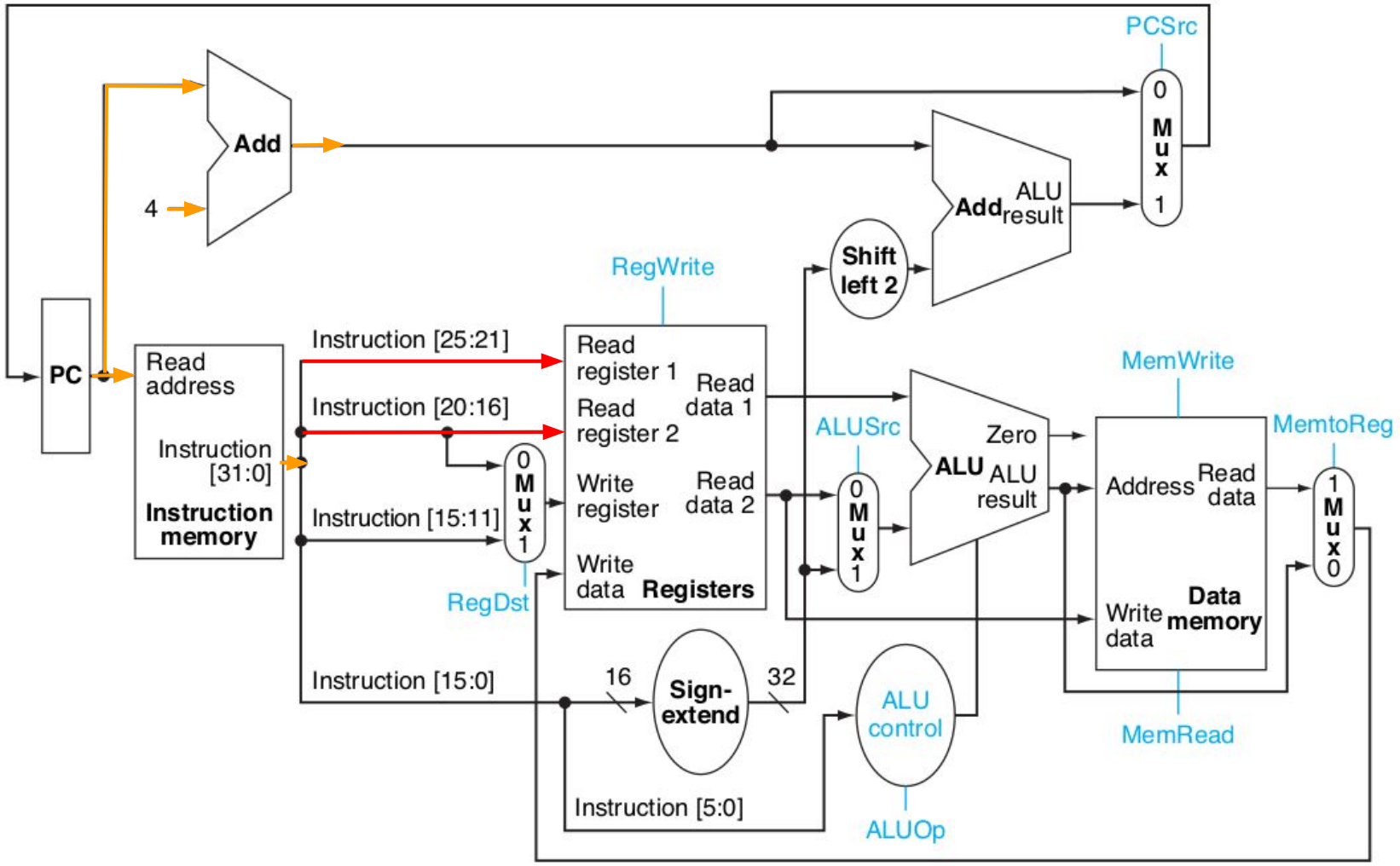


FIGURE 4.15

Instruções Formato R - Leitura Reg

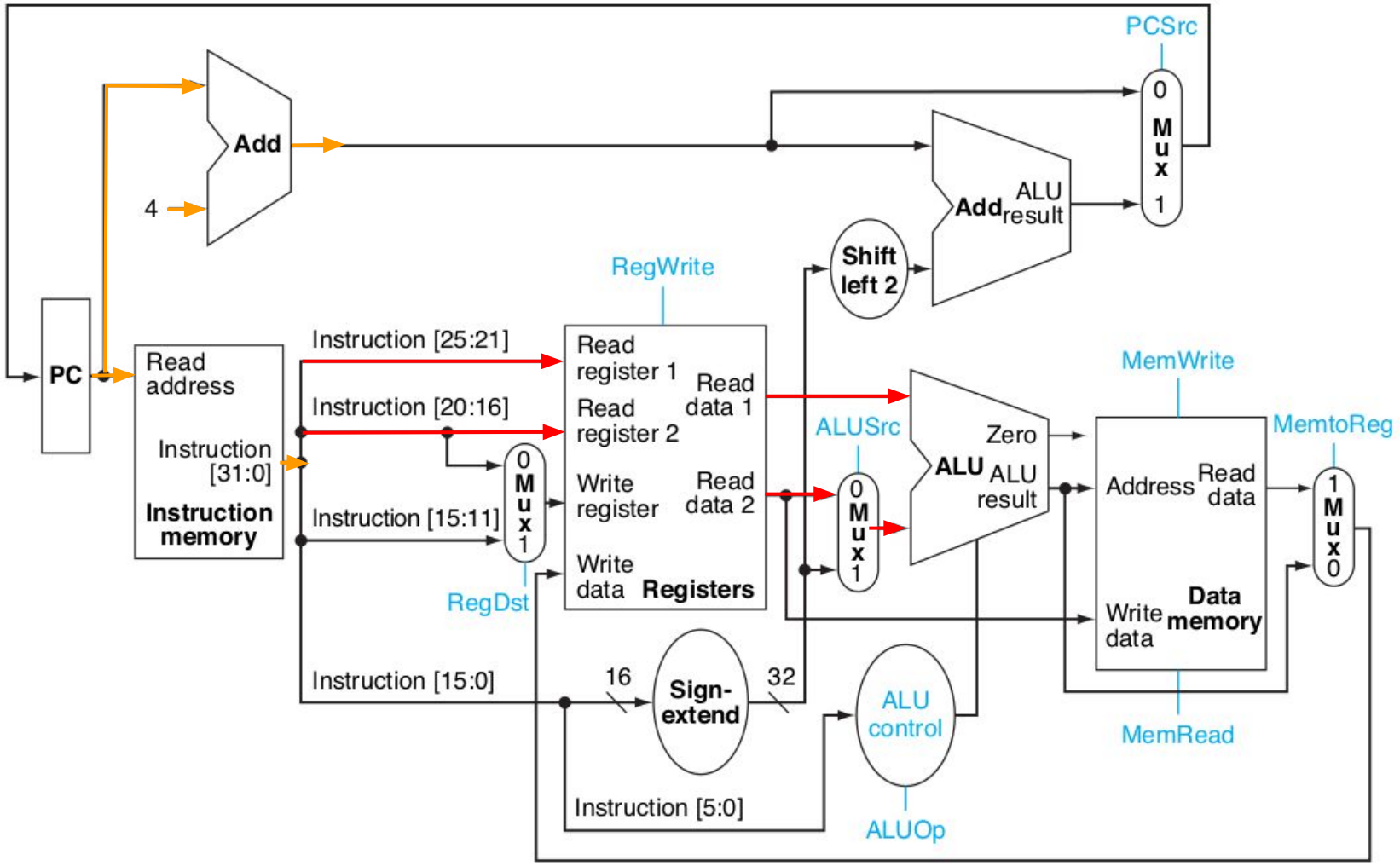


FIGURE 4.15

Instruções Formato R - Cálculo

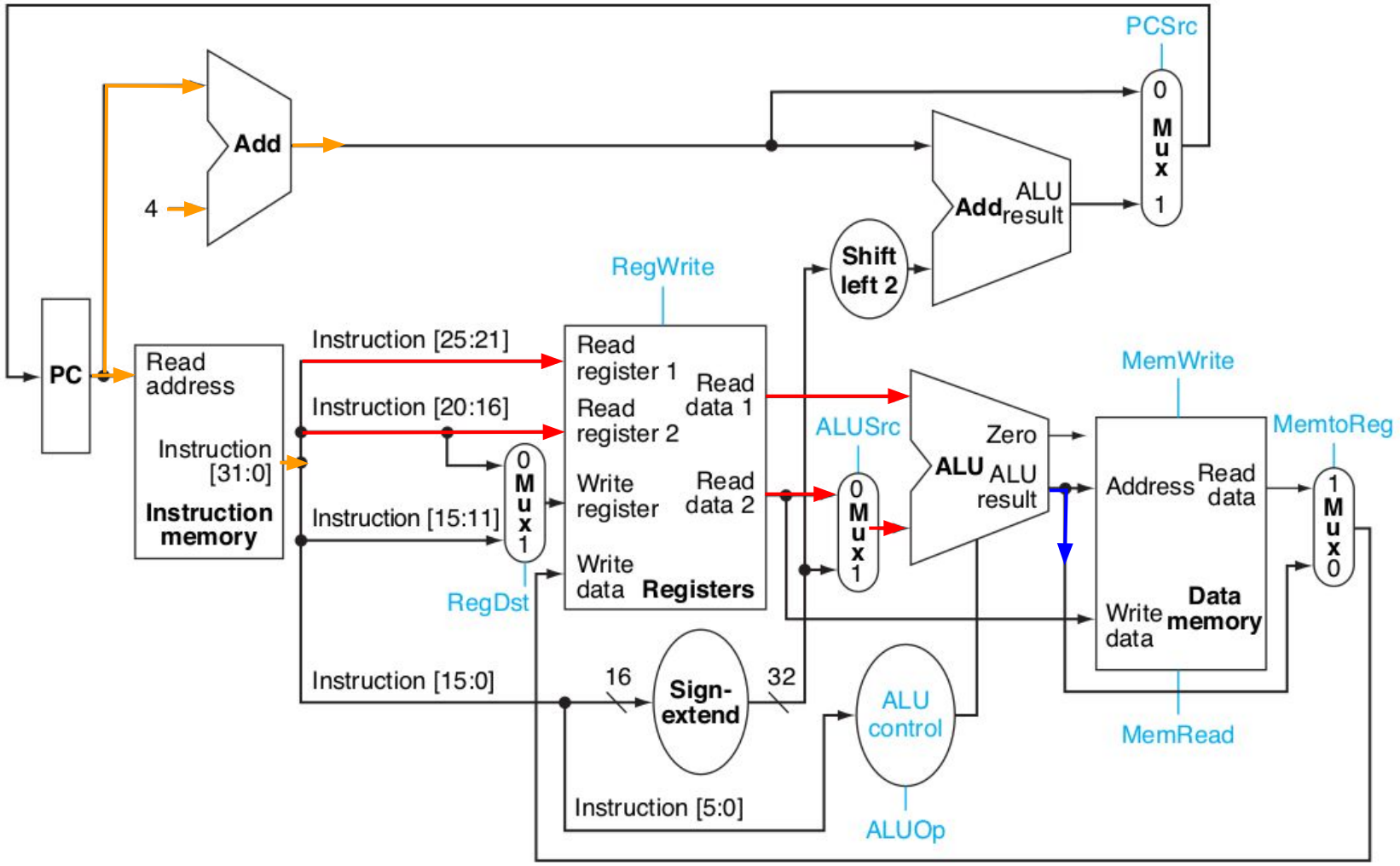


FIGURE 4.15

Instruções Formato R - Escrita Reg

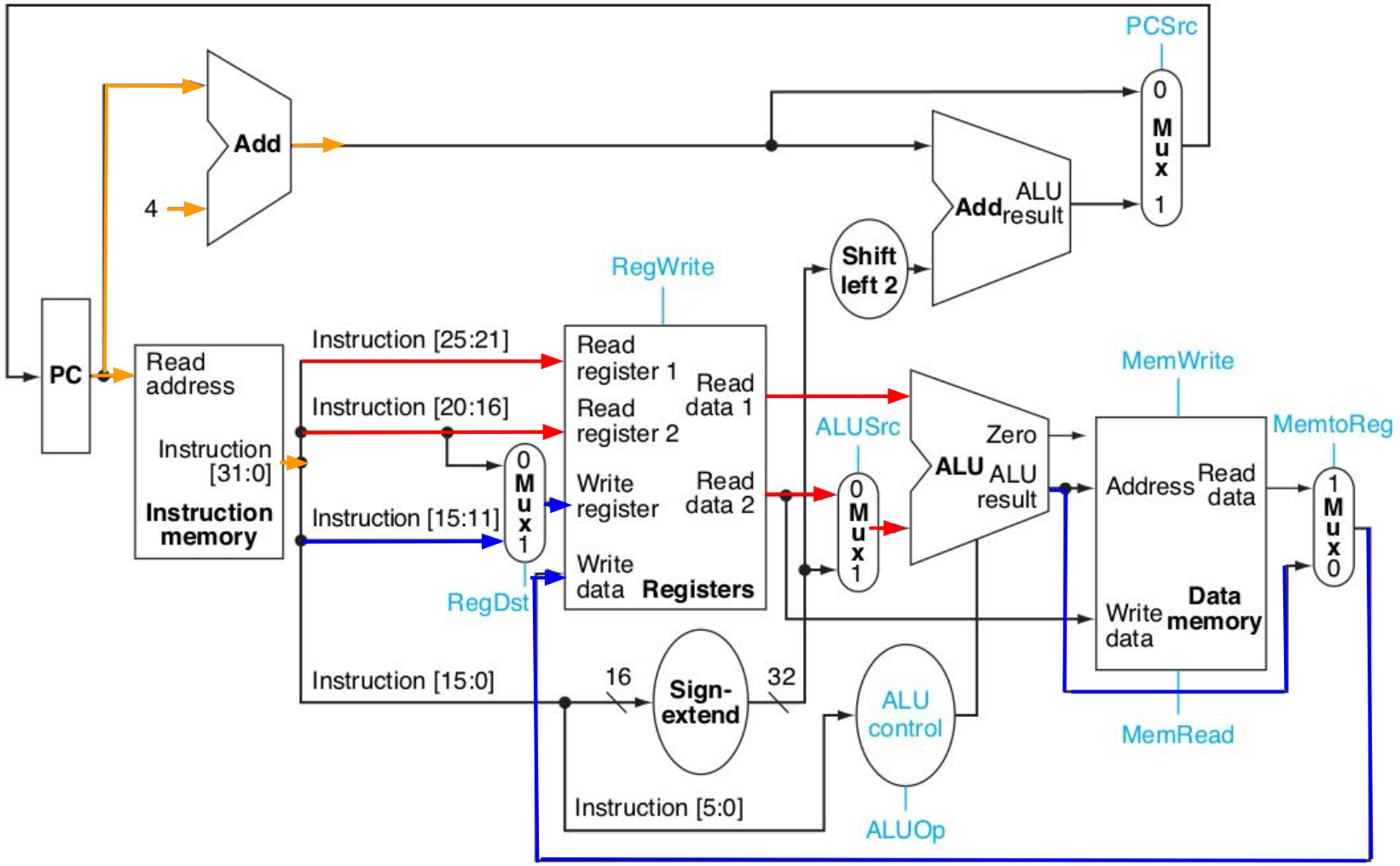
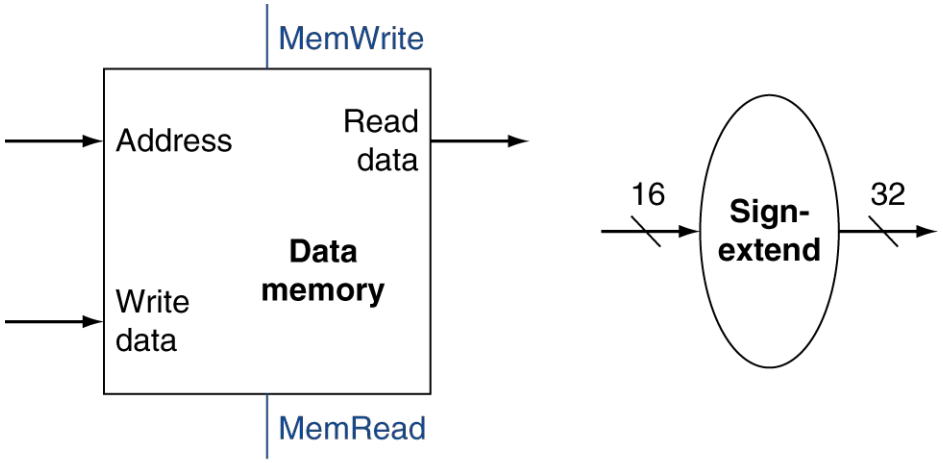


FIGURE 4.15

Instruções Load/Store

- Lê operandos dos registradores
- Calcular endereço usando offset de 16 bits
  - Usa ULA, mas estende o sinal do offset
- Load: Lê valor da memória e escreve no registrador
- Store: Lê valor do registrador e escreve na memória



a. Data memory unit

b. Sign extension unit

Instruções Load/Store

0x00400018 lw $t0, 0($t1) (lw $8,0x00000000($9))																R[rt] = M[R[rs]+SignExtImm] (23 hex)															
0x23						9						8																0x0000			
1	0	0	0	1	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8				D				2				8				0				0				0				0			
R[rt] = M[R[rs]+SignExtImm] R[8] = M[R[9]+0x0000.0000]																													

Instruções Load - Busca

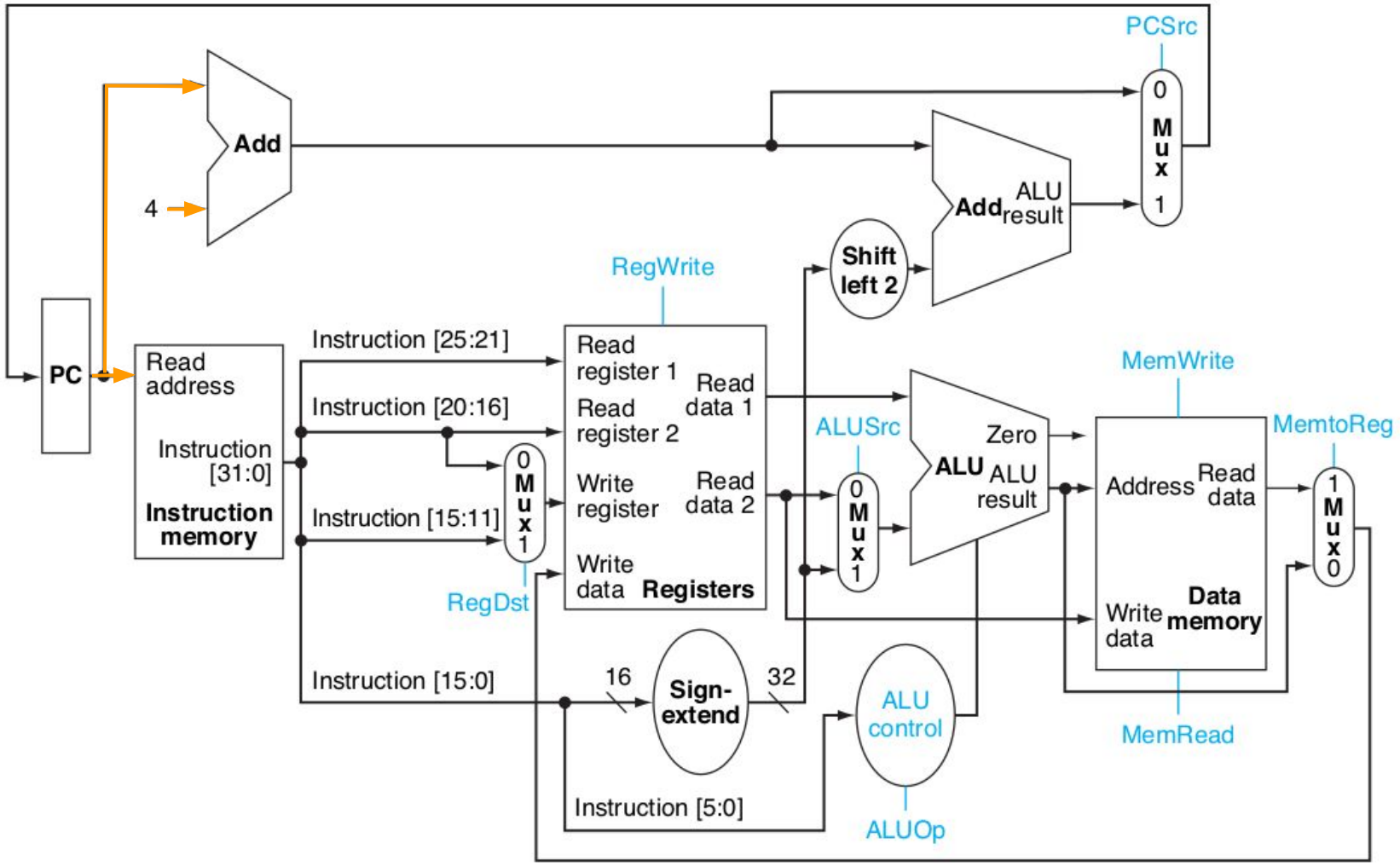


FIGURE 4.15

Instruções Load - Busca

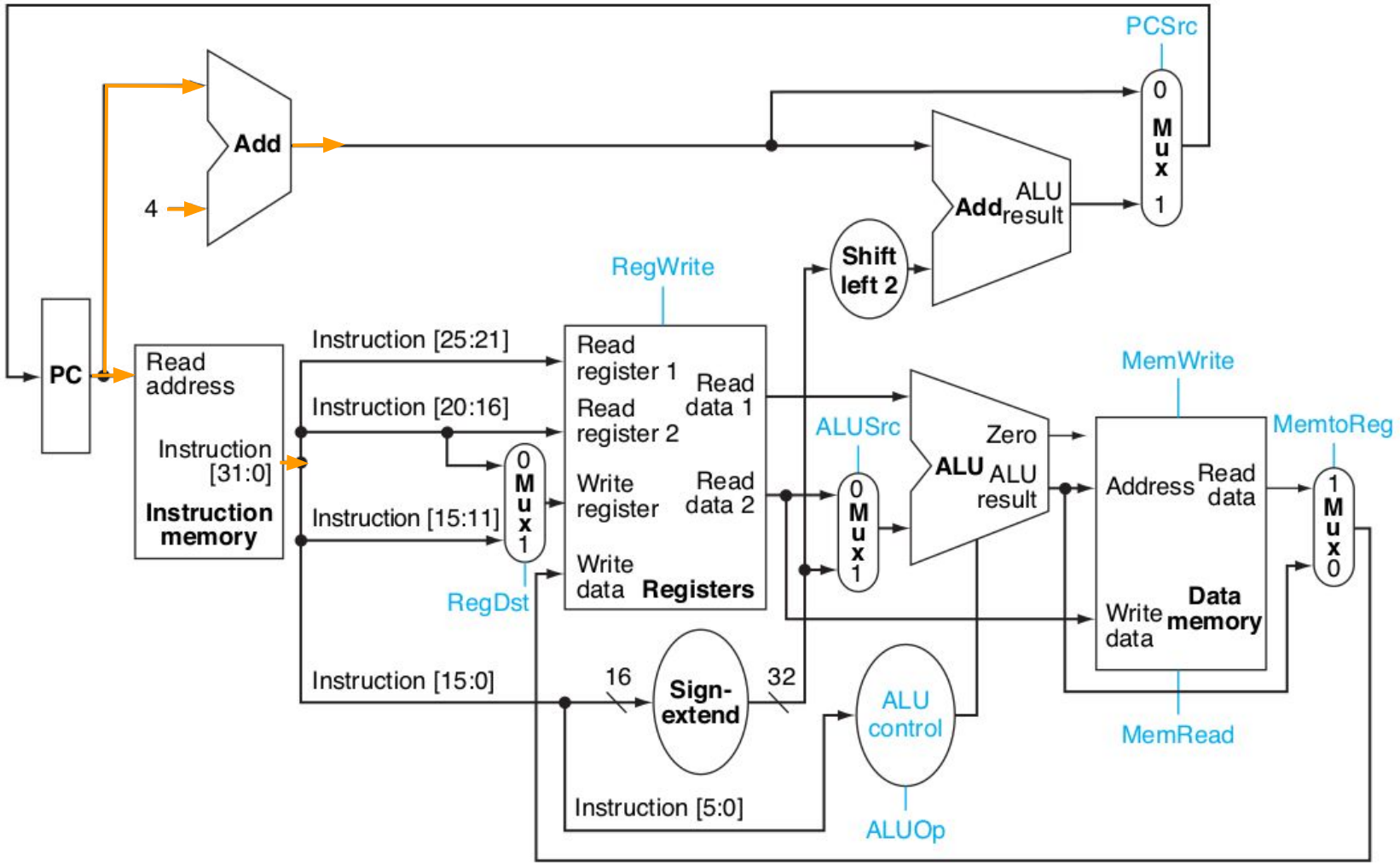
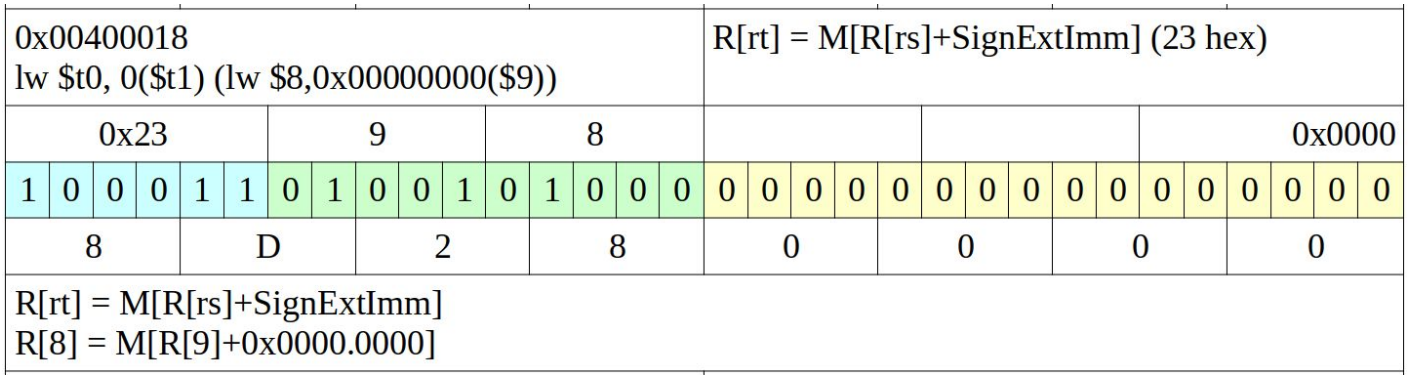
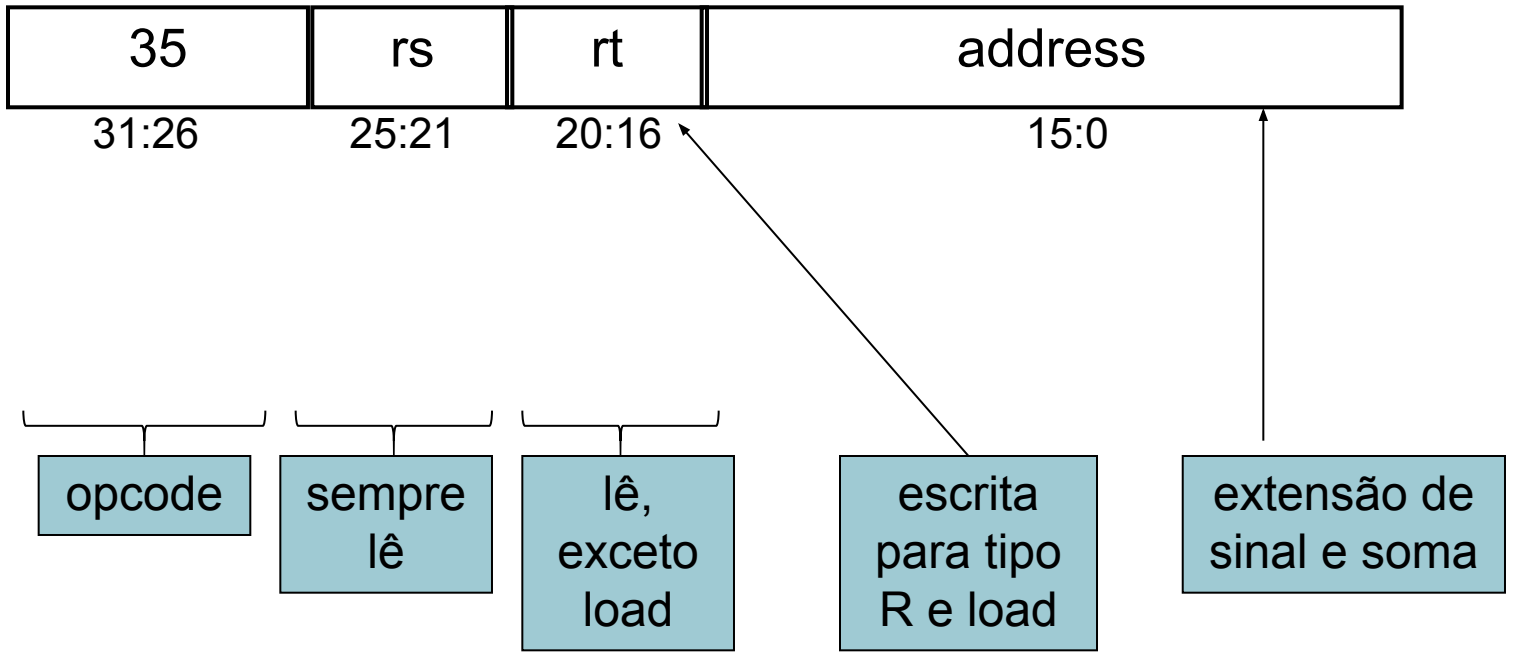


FIGURE 4.15

2 - Leitura de Operandos



Load



Instruções Load - Leitura Reg

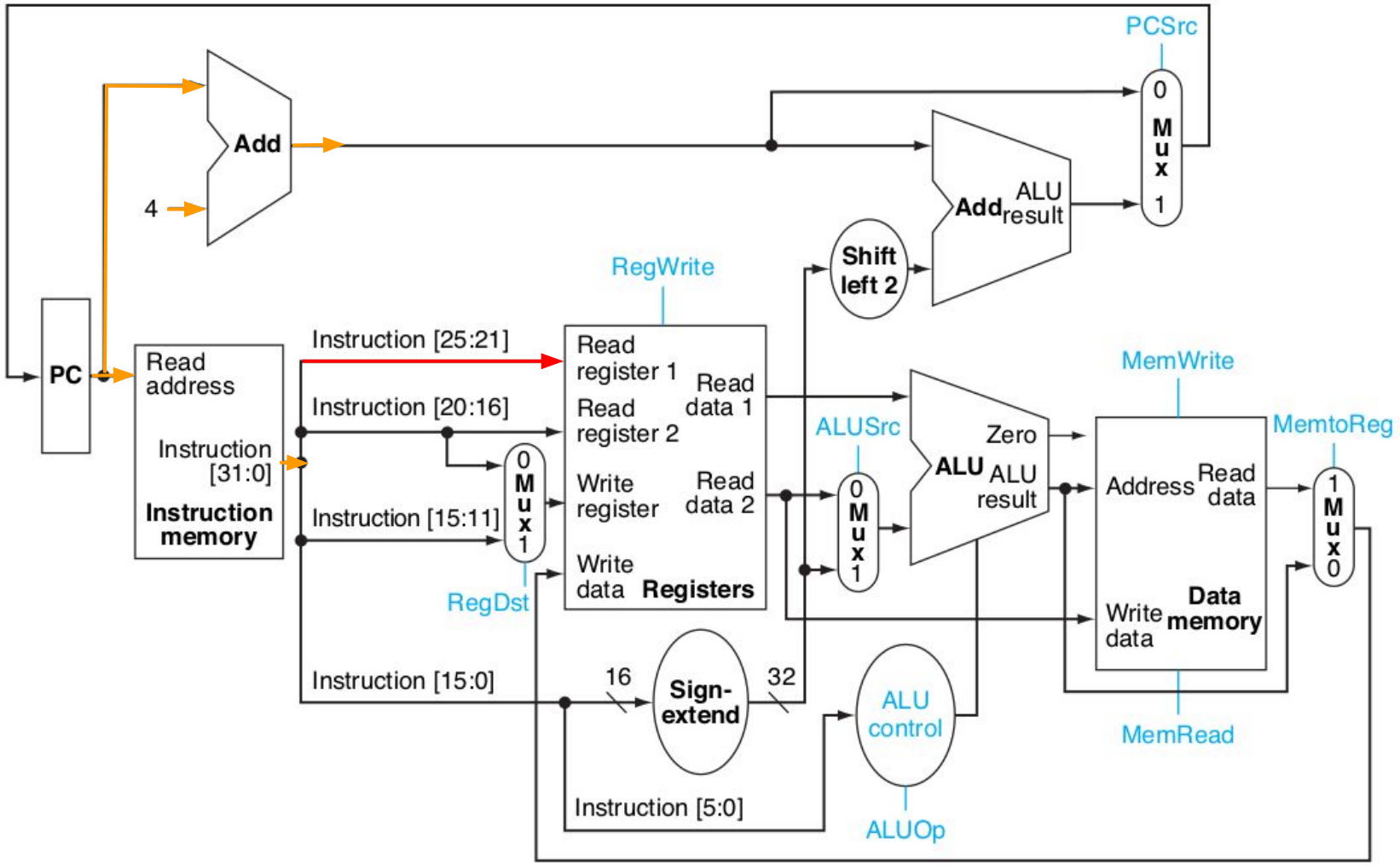


FIGURE 4.15

Instruções Load - Leitura Reg

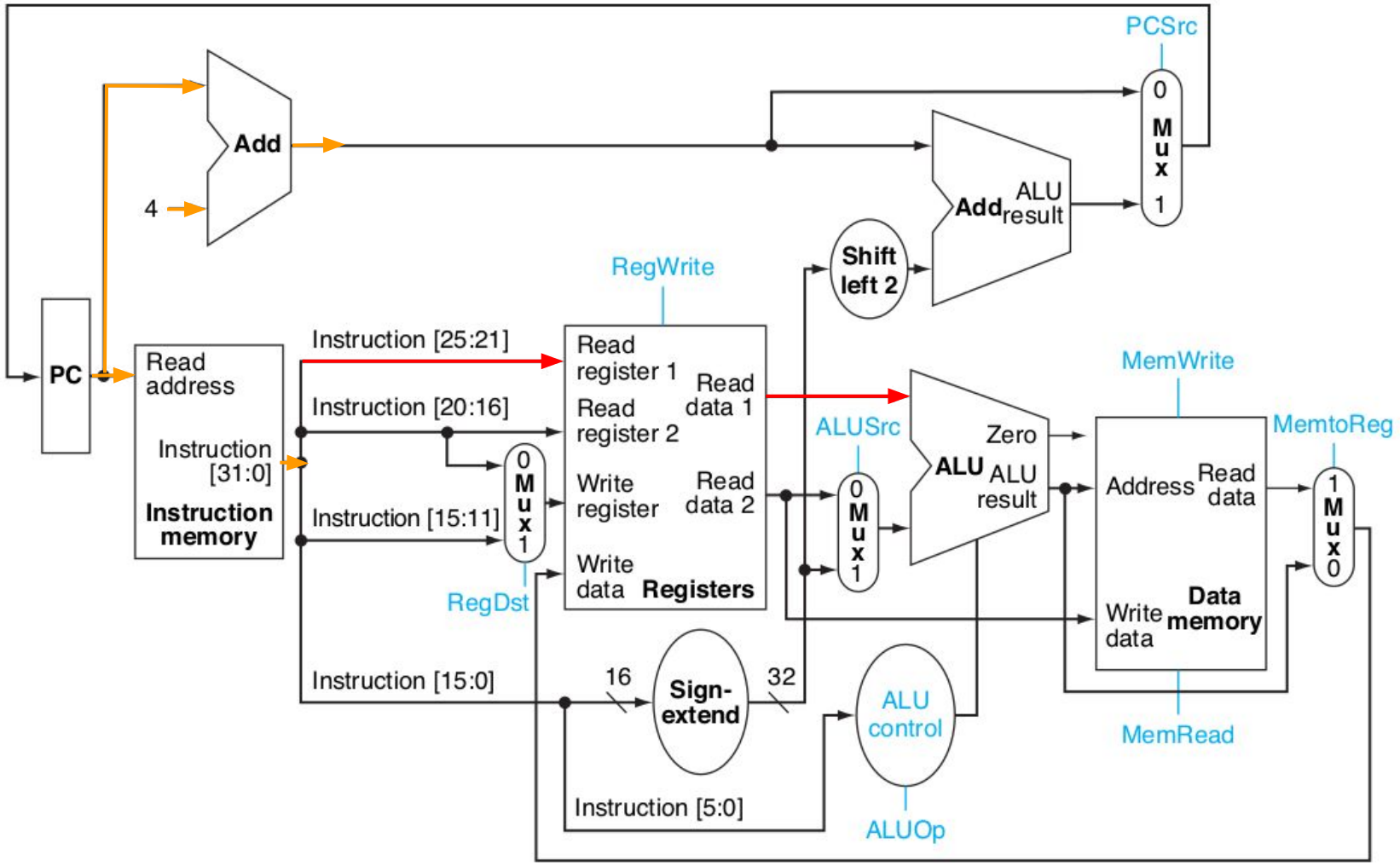


FIGURE 4.15

Instruções Load - Cálculo End

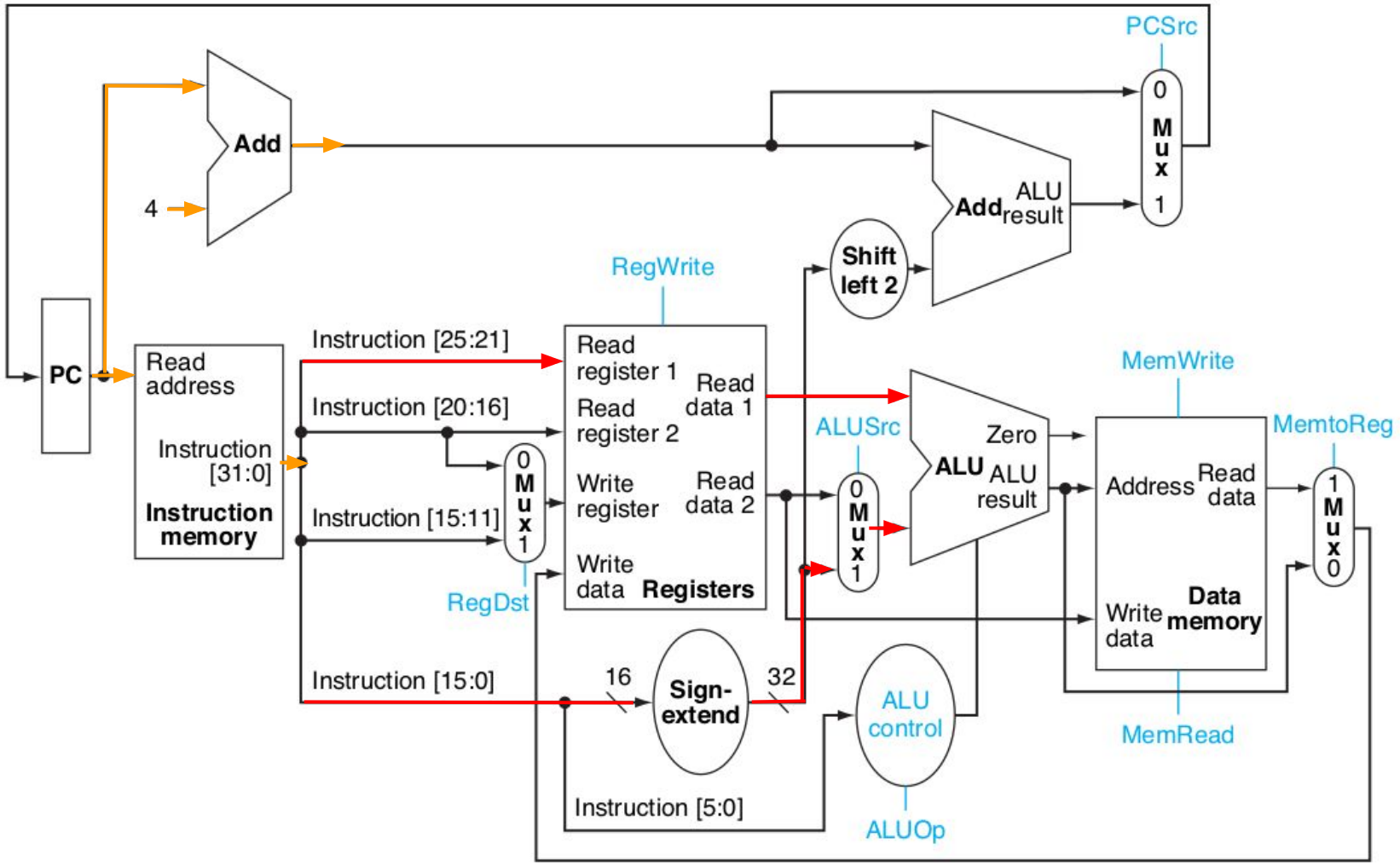


FIGURE 4.15

Instruções Load - Cálculo End

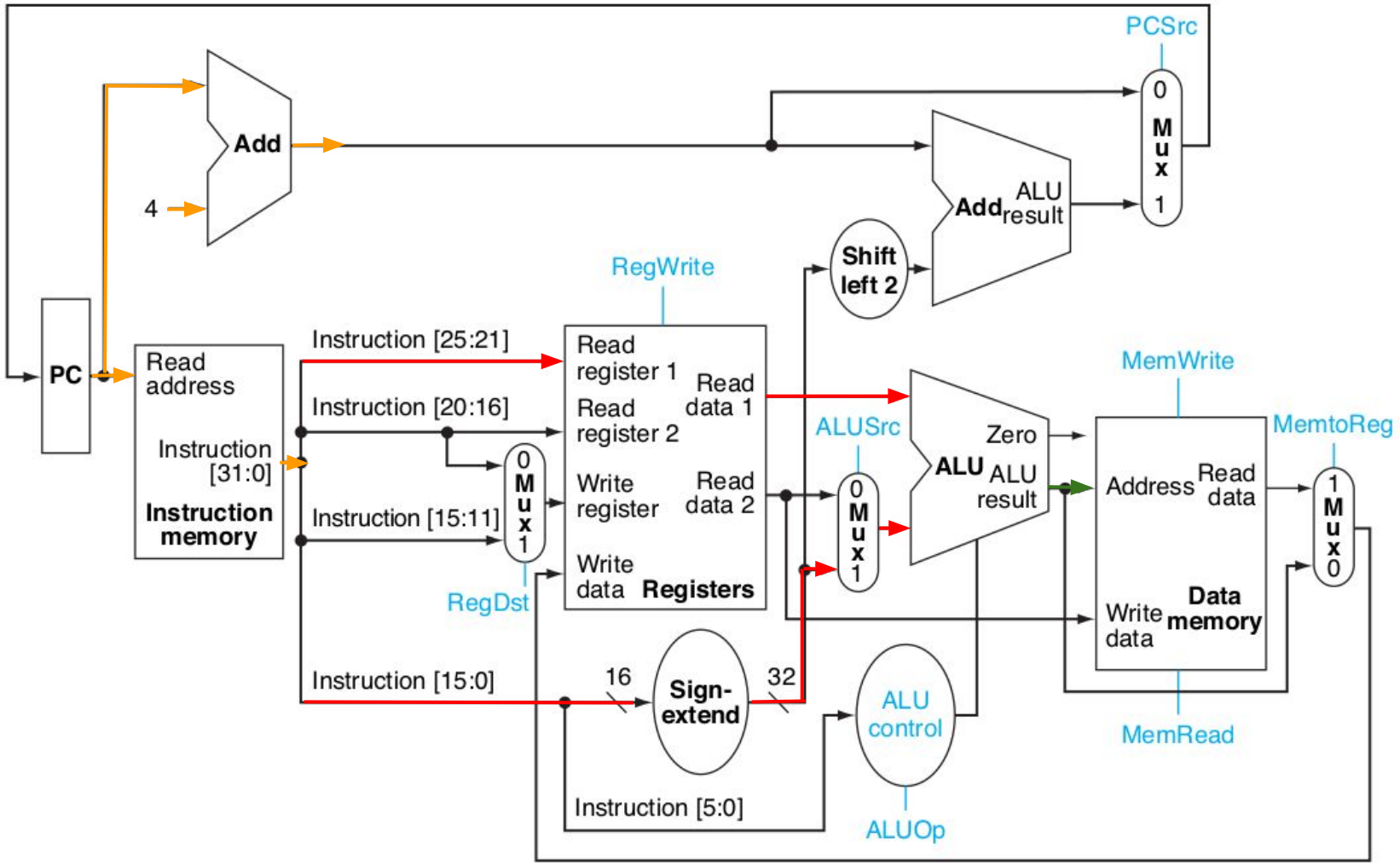


FIGURE 4.15

Instruções Load - Leitura Mem

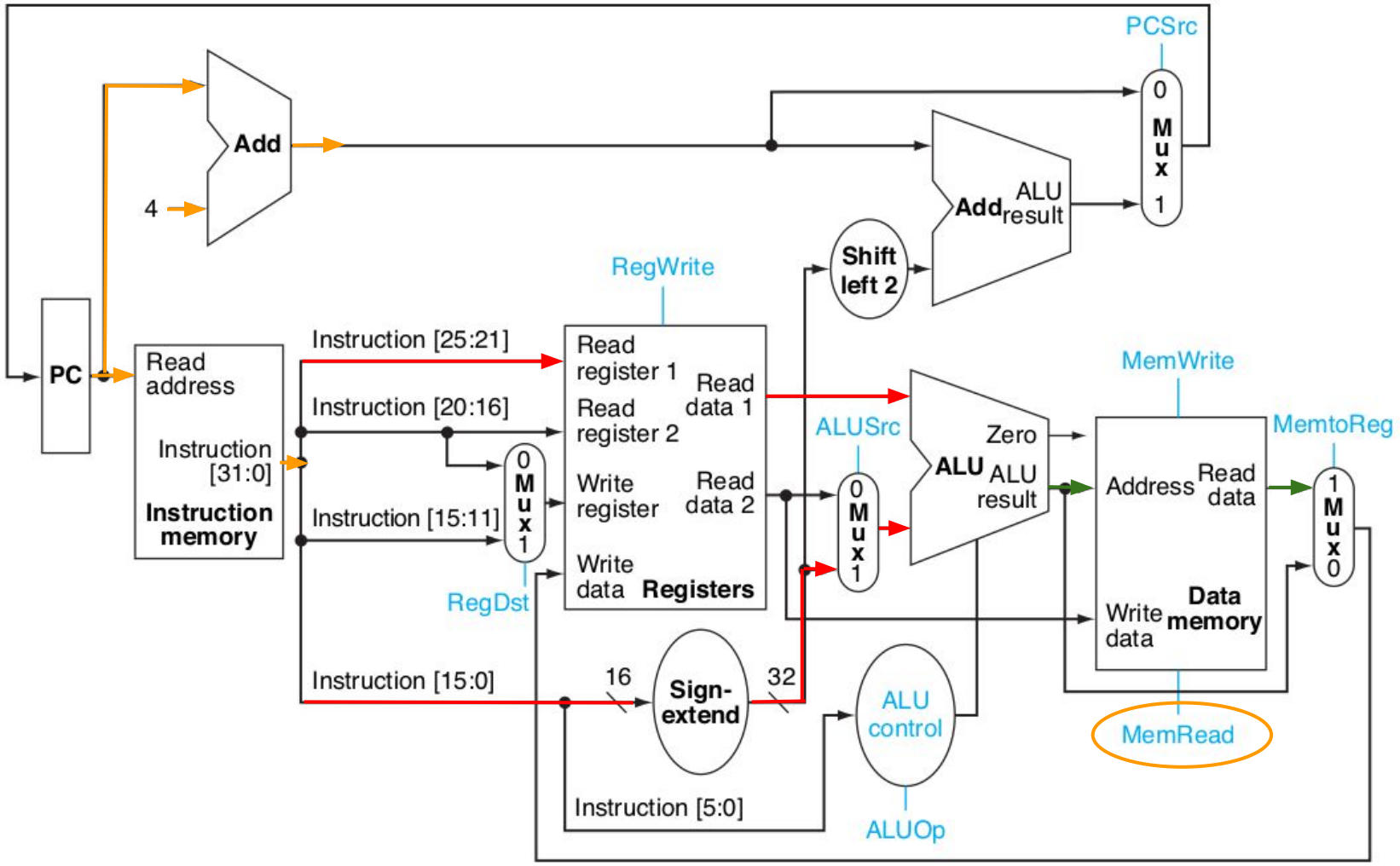


FIGURE 4.15

Instruções Load - Escrita Reg

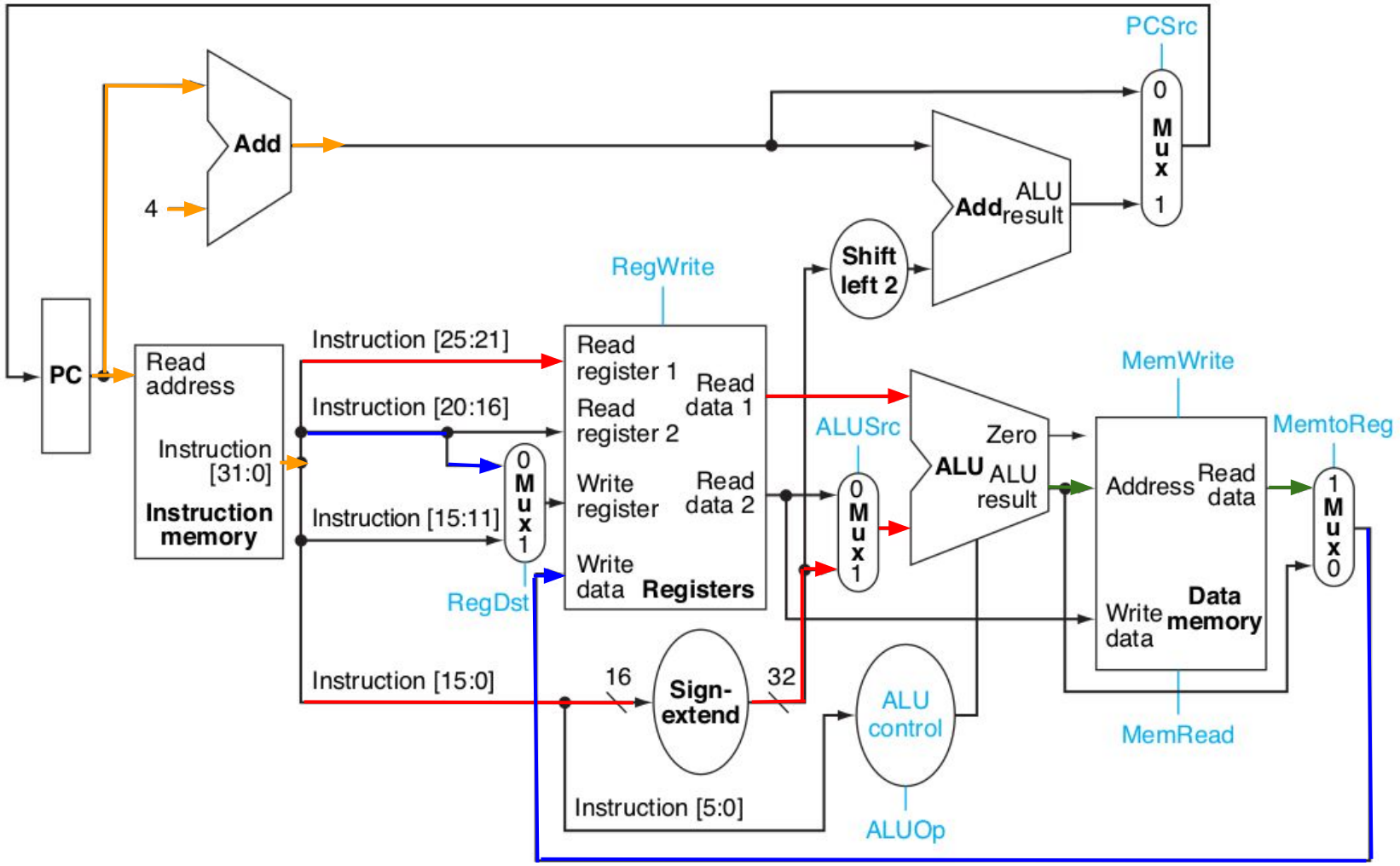


FIGURE 4.15

Instruções Store - Busca

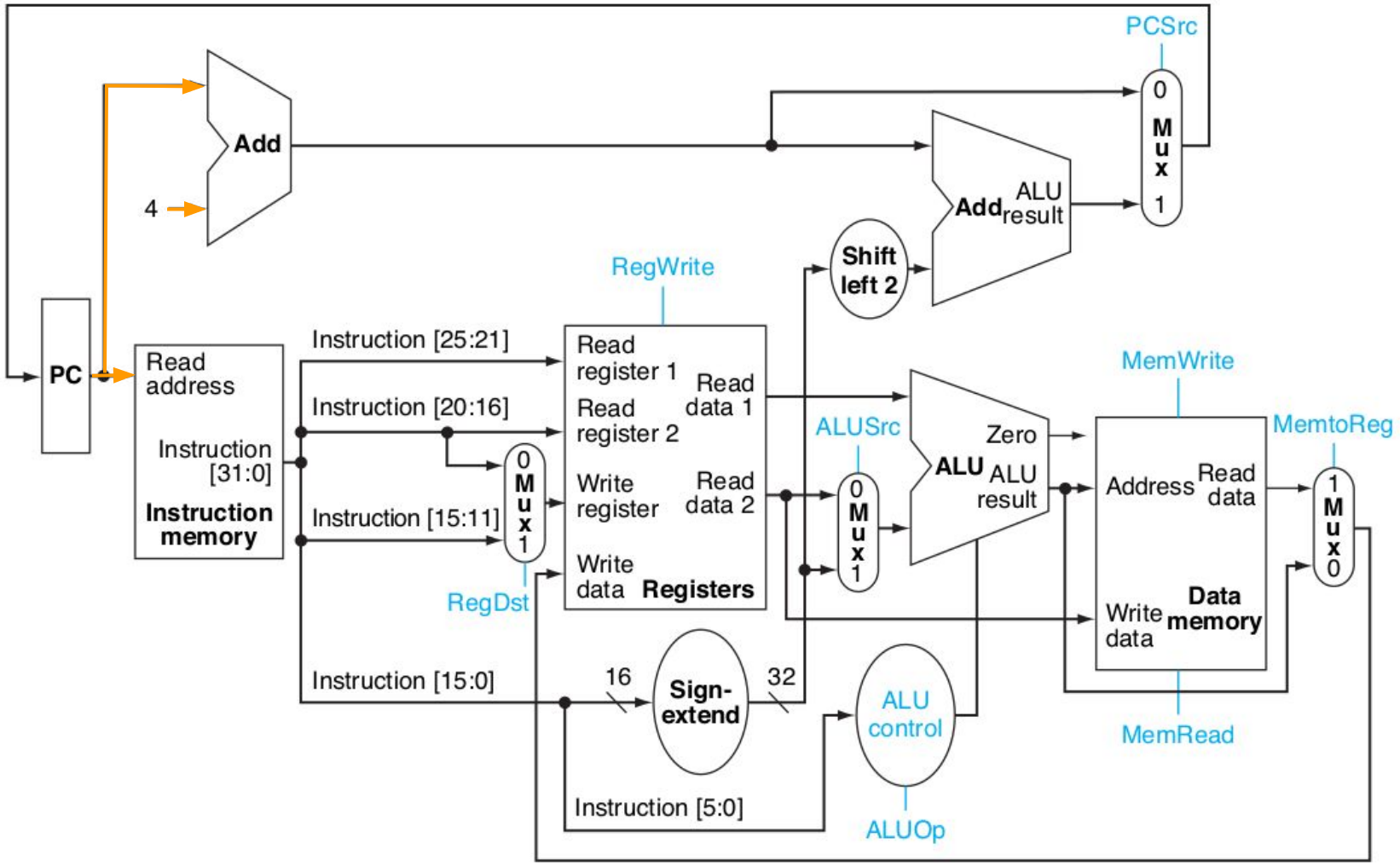


FIGURE 4.15

Instruções Store - Busca

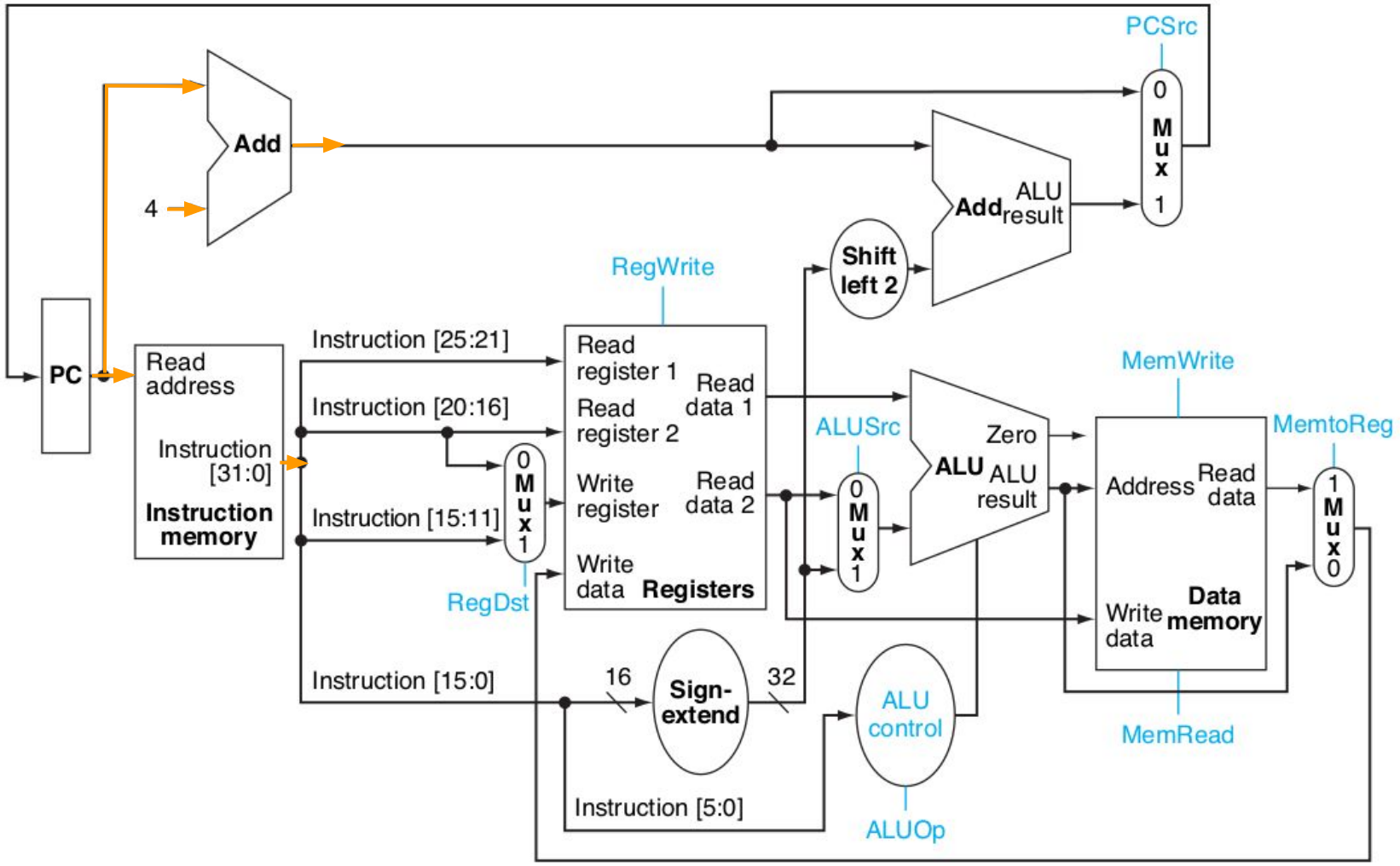
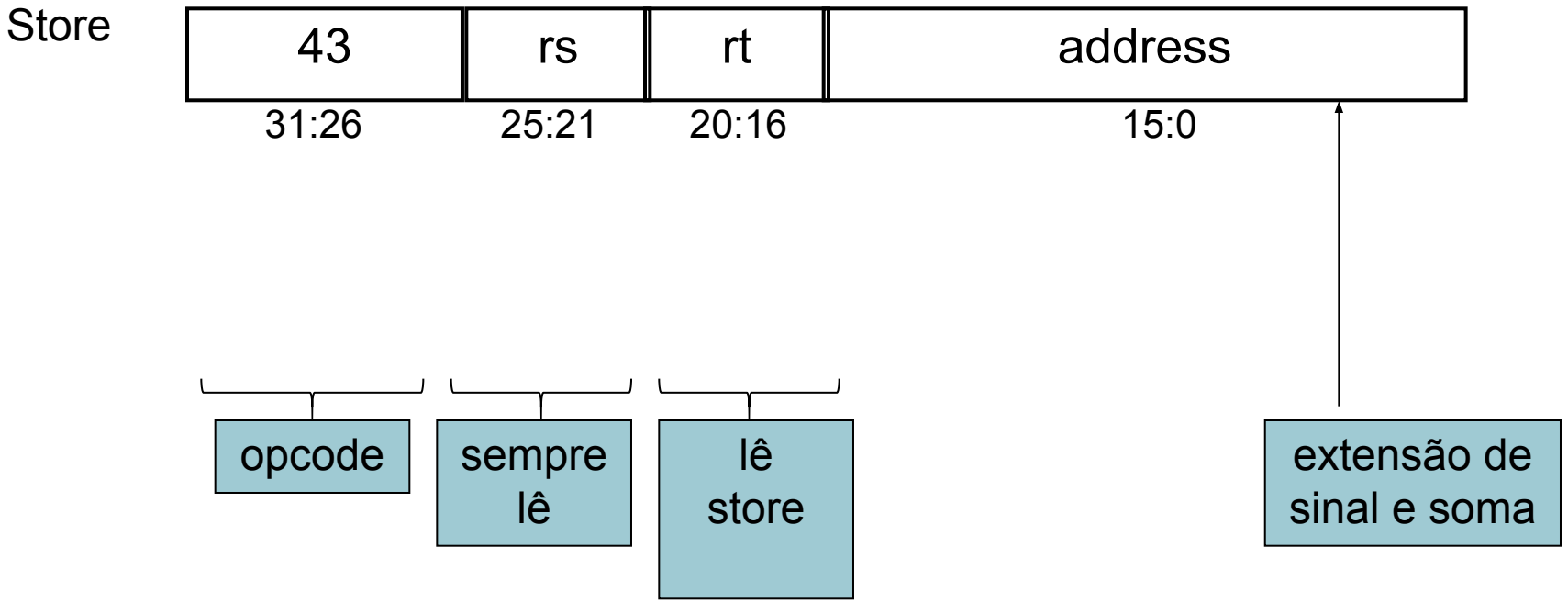


FIGURE 4.15

2 - Leitura de Operandos



Instruções Store - Leitura Reg

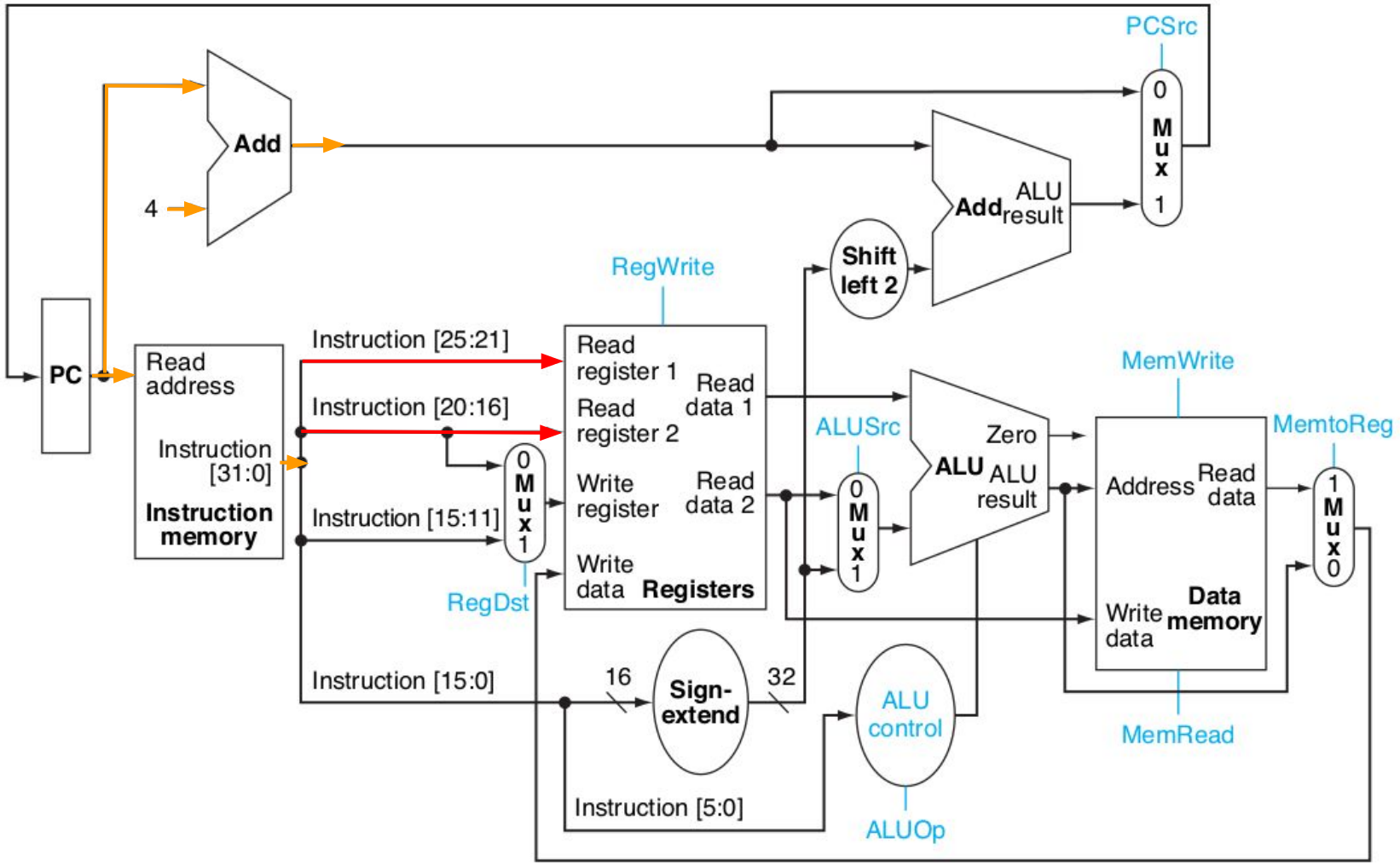


FIGURE 4.15

Instruções Store - Leitura Reg

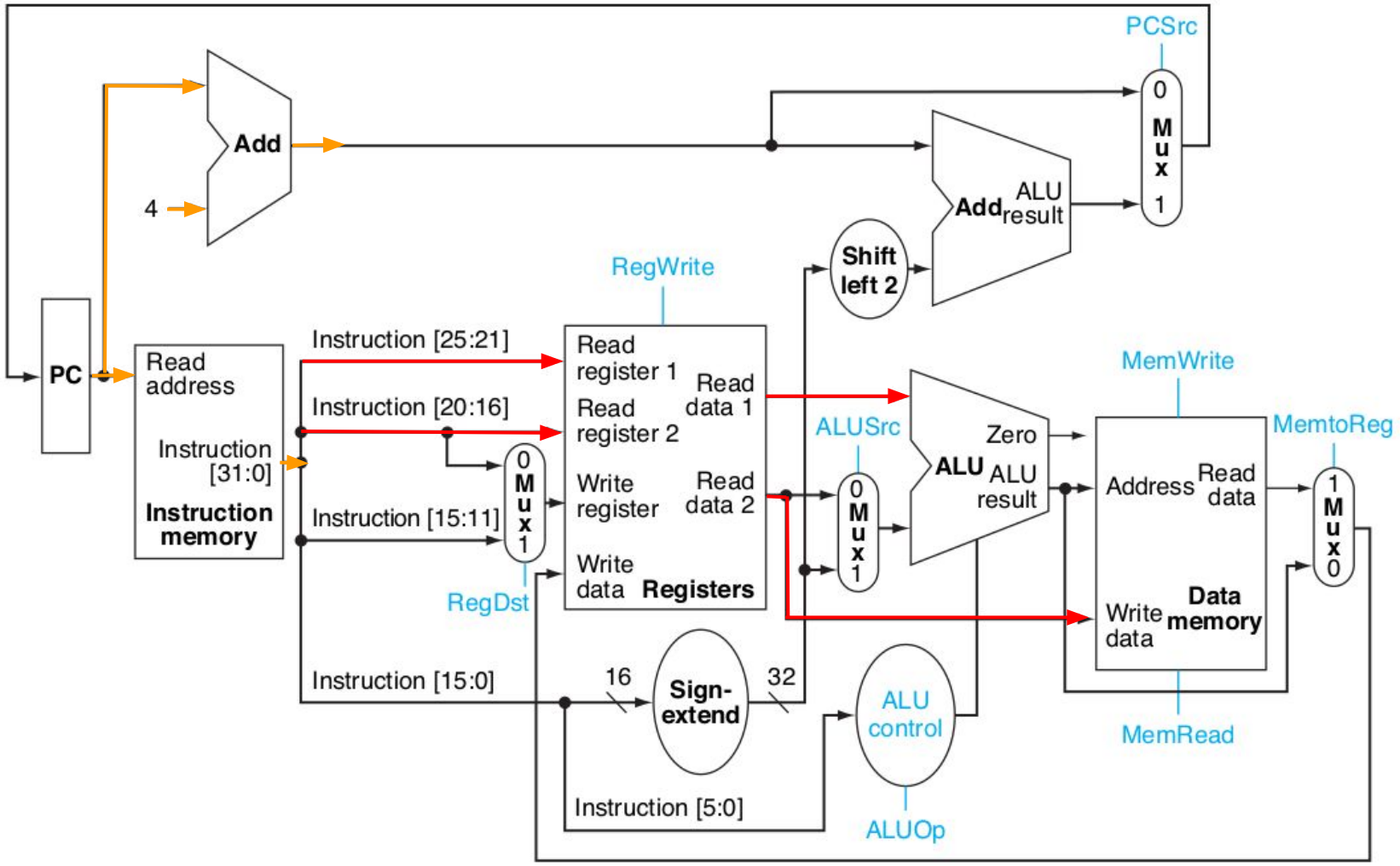


FIGURE 4.15

Instruções Store - Cálculo End

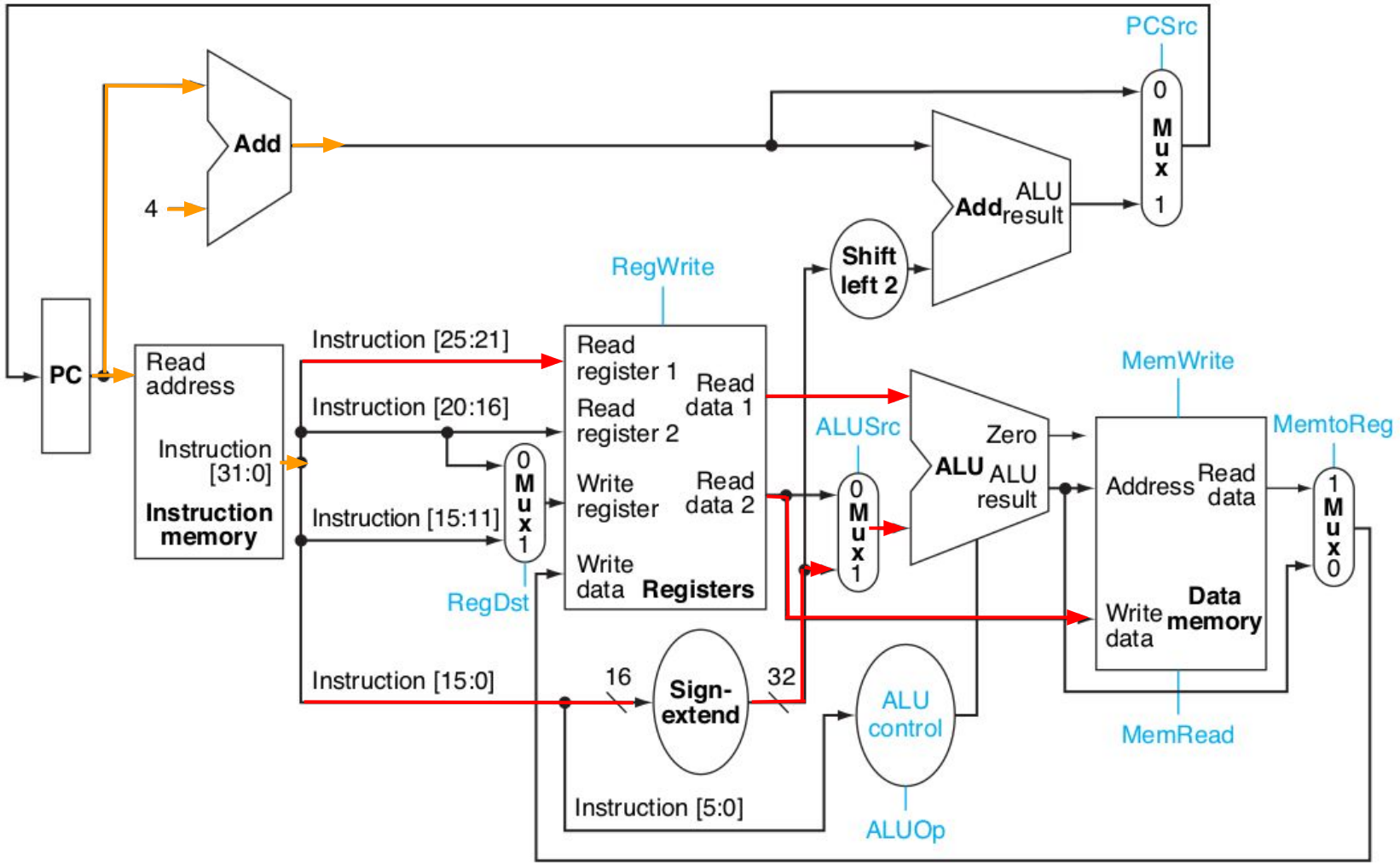


FIGURE 4.15

Instruções Store - Cálculo End

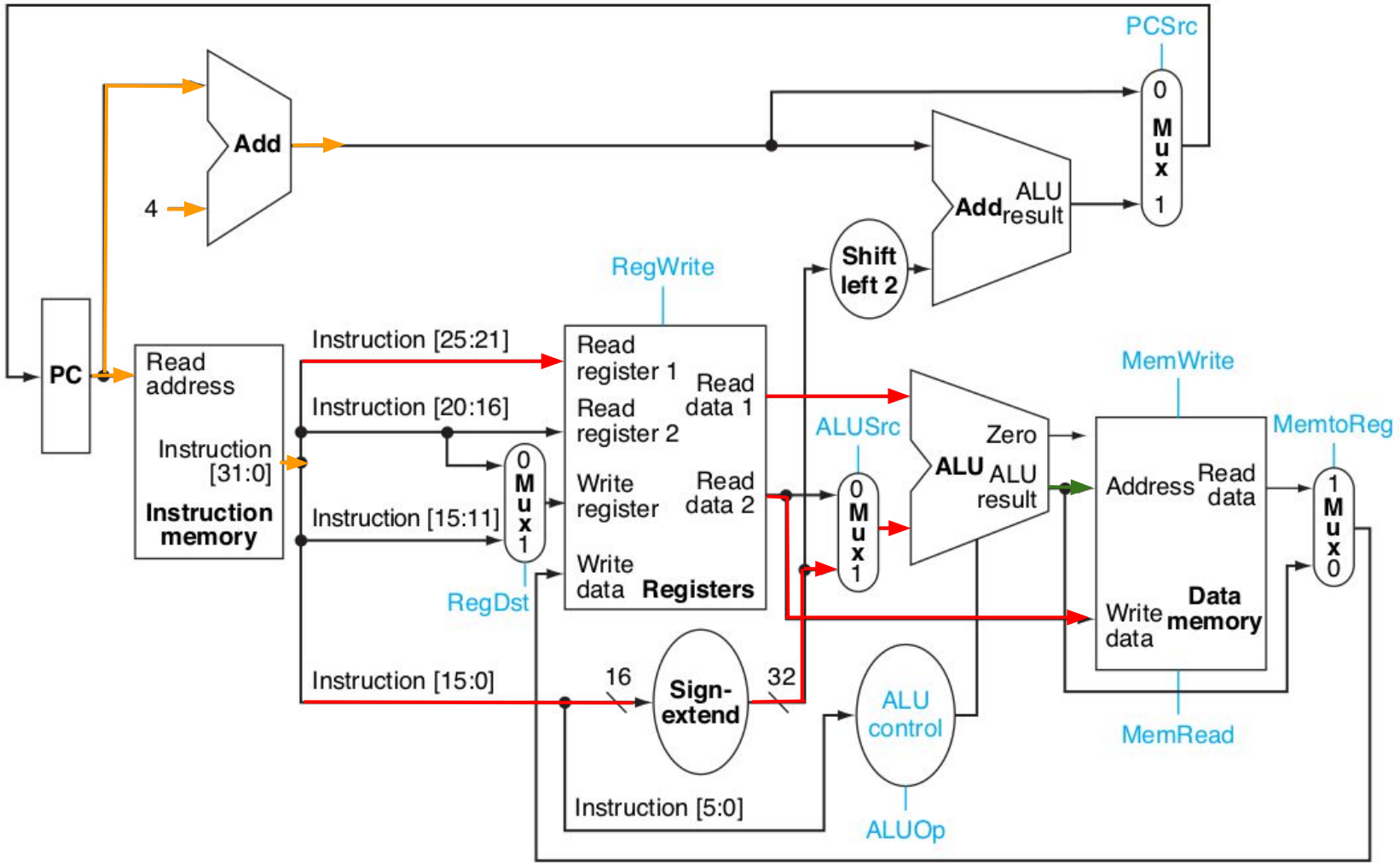


FIGURE 4.15

Instruções Store - Escrita Mem

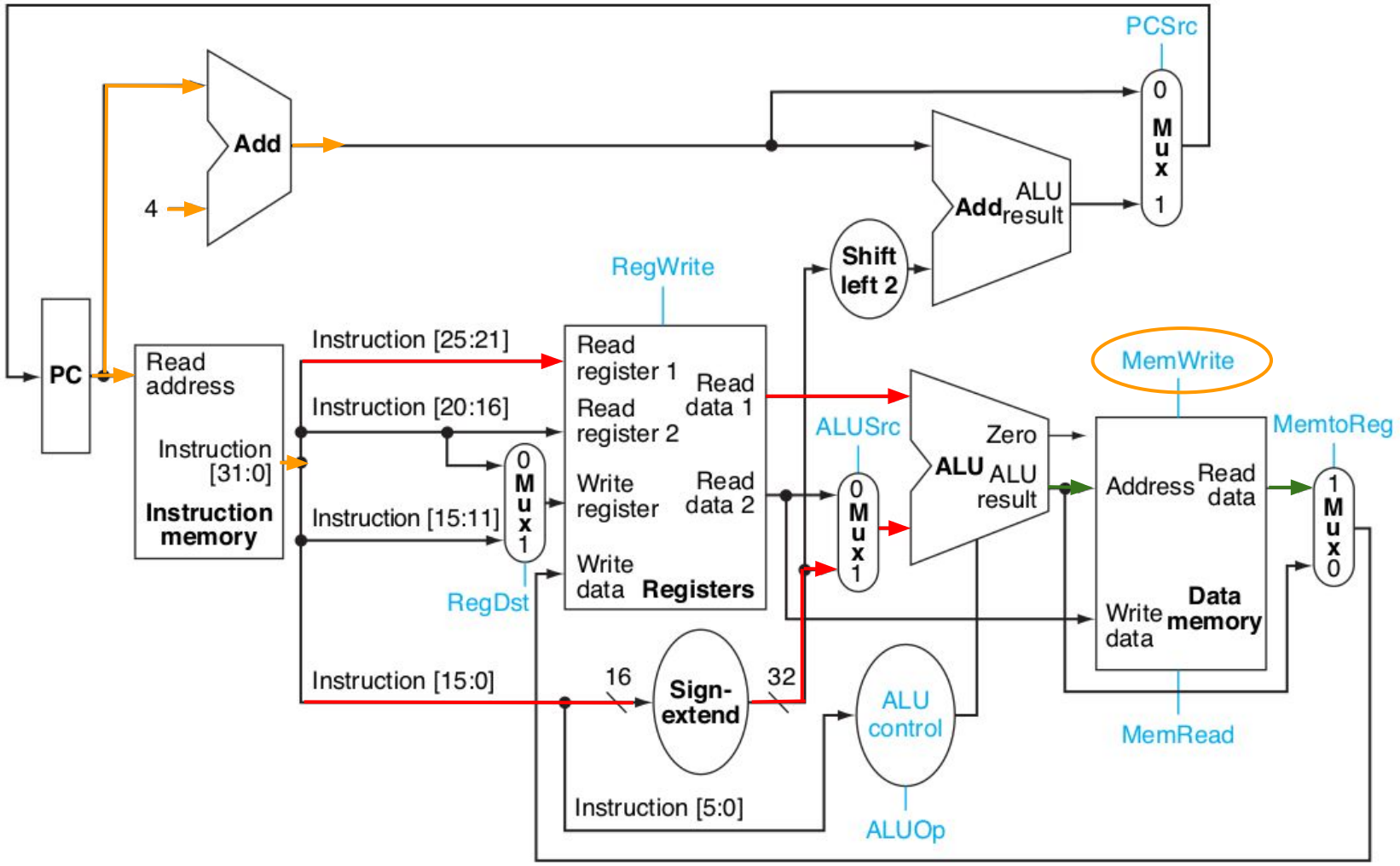


FIGURE 4.15

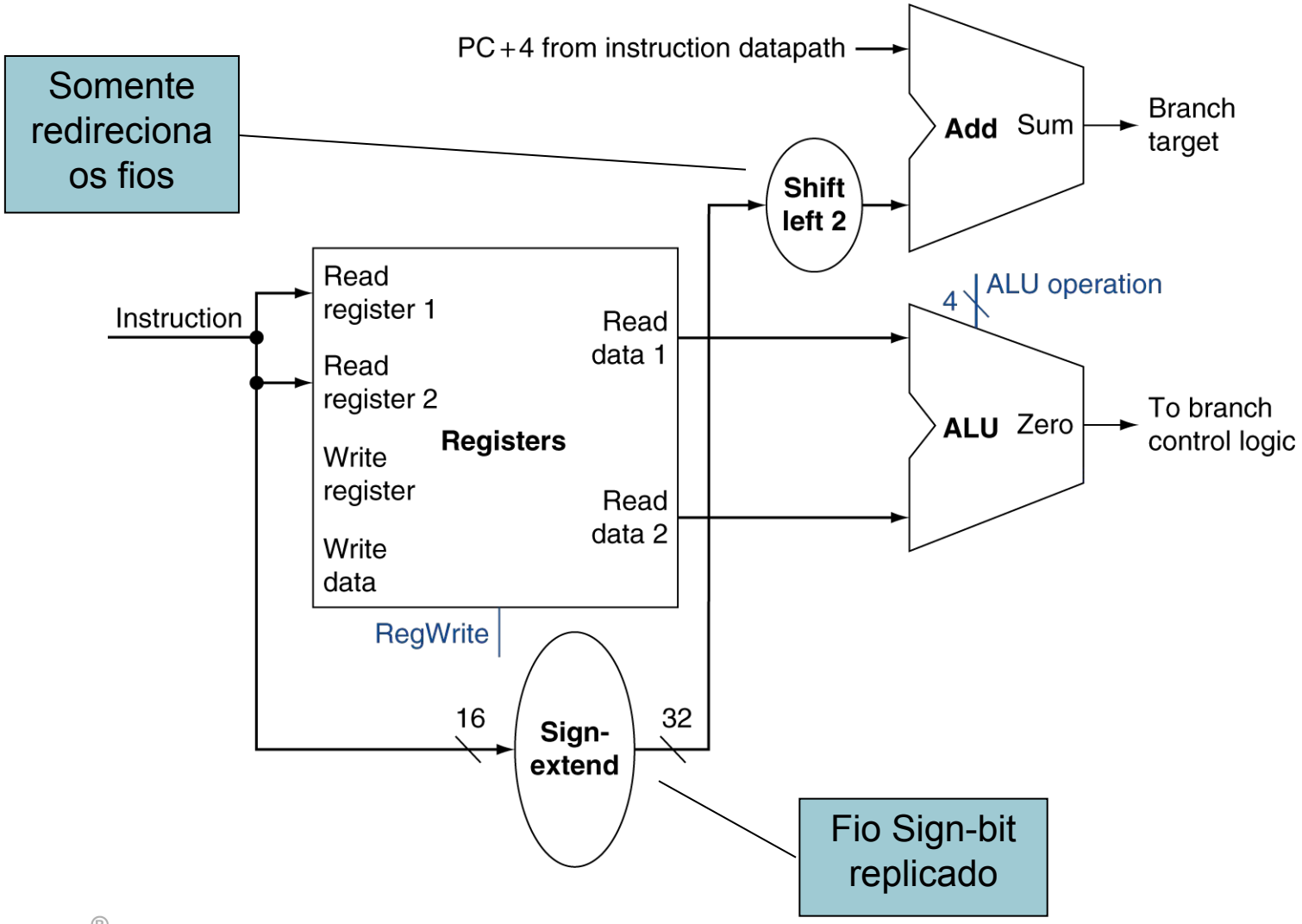
Instruções de Desvio

- Lê operandos dos registradores
- Compara os operandos
  - Usa ULA, subtrai e verifica saída Zero
- Calcula endereço de destino
  - Estende o sinal do offset
  - Desloca 2 bits à esquerda (offset palavra)
  - Adiciona com PC + 4
    - Já calculado pela busca da instrução

Instruções de Desvio

0x0040001c bne $t0, $s5, Exit (bne $8,$21,0x00000002)																if(R[rs]!=R[rt]) PC=PC+4+BranchAddr (0x05)														
0x05						8					21																				
0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1				5				1				5				0				0				0				2		
0x0002 extendido sinal de 16 bits para 32 bits = 0x0000.0002 0x0000.0002 << 2 = 0x0000.0008 bytes PC = (PC+4) +BranchAddr PC = 0x00400020 + 0x00000008 = 0x00400028 (Exit: nop)																														

Instruções de Desvio



Instruções Desvio - Busca

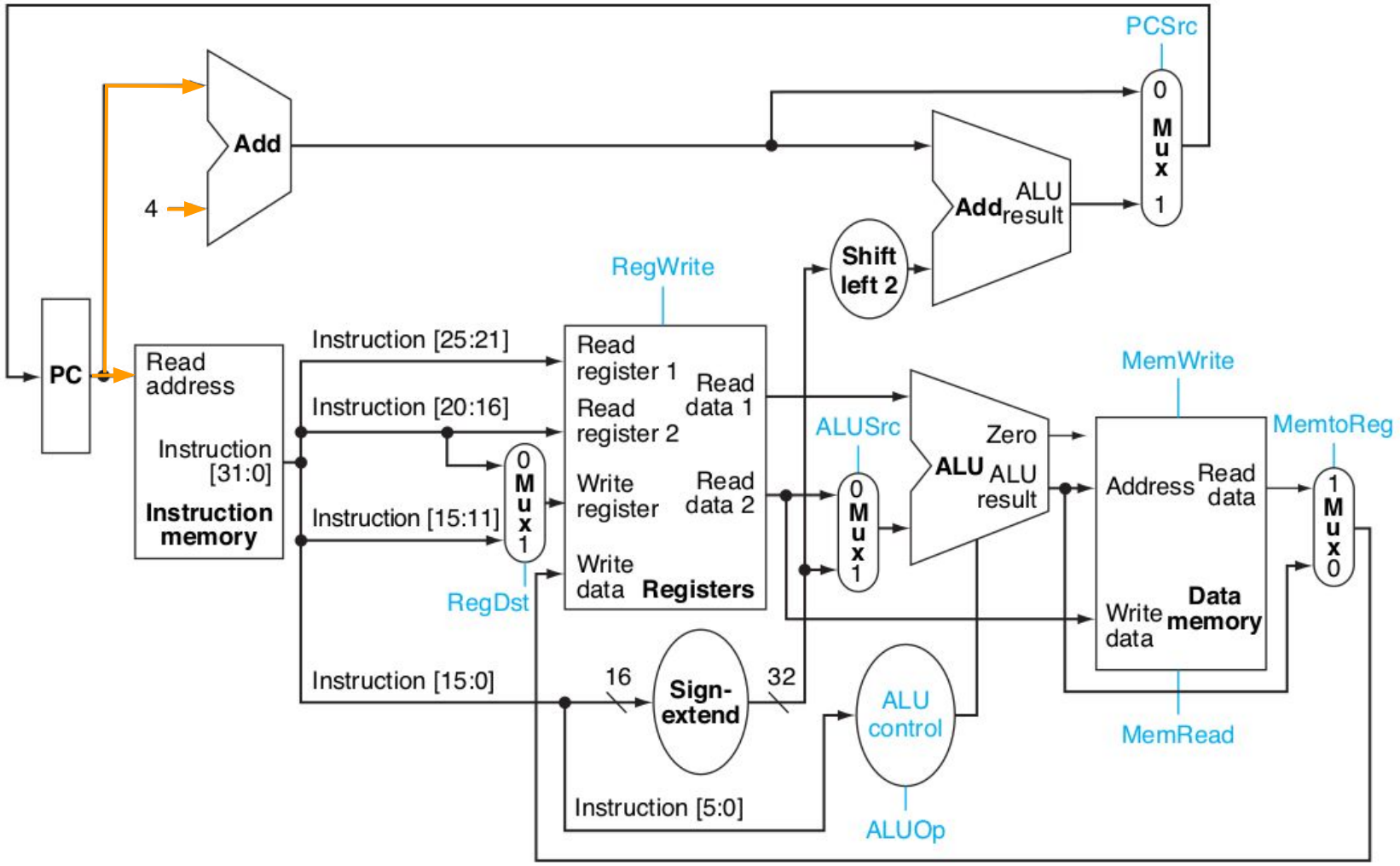


FIGURE 4.15

Instruções Desvio - Busca

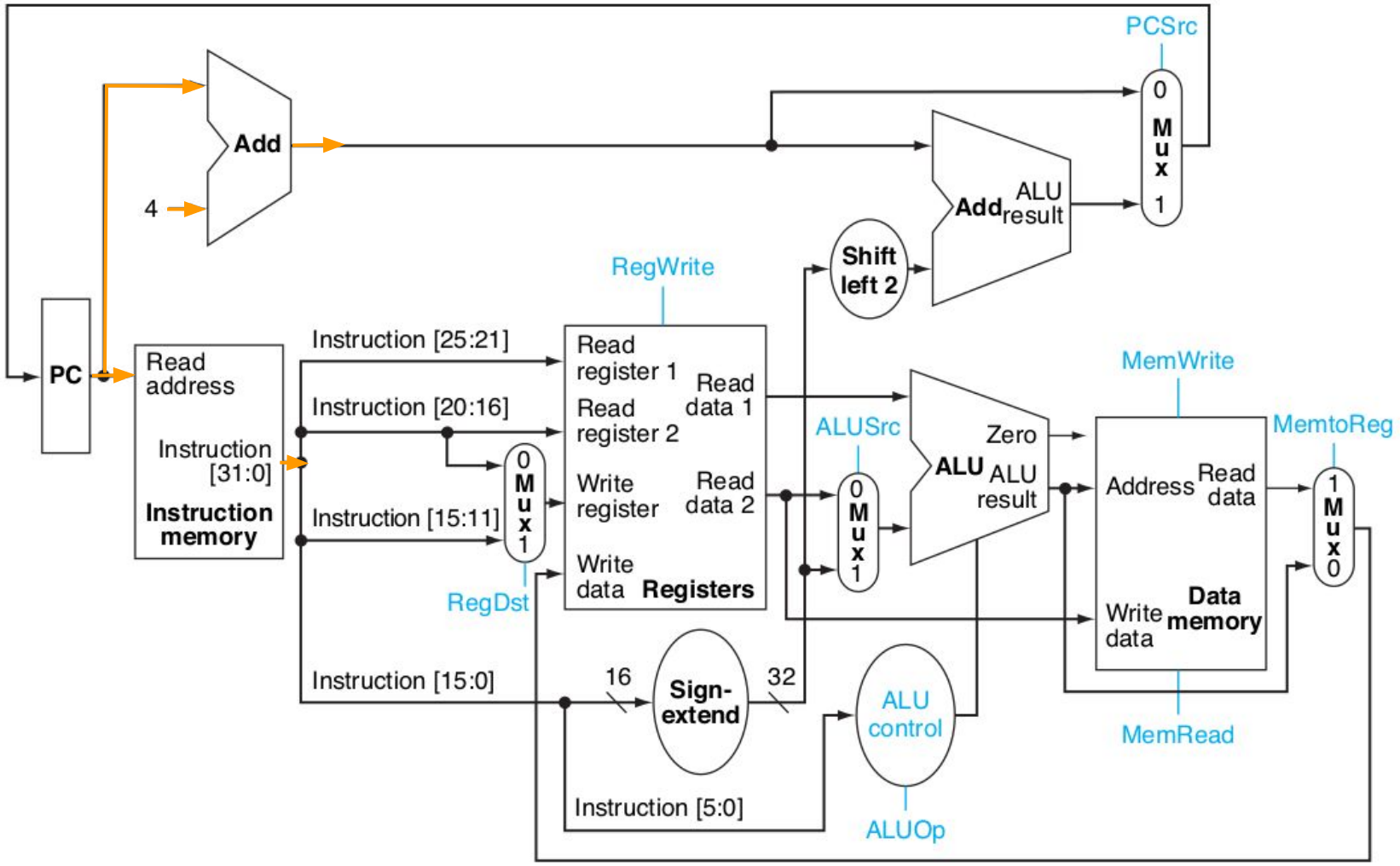


FIGURE 4.15

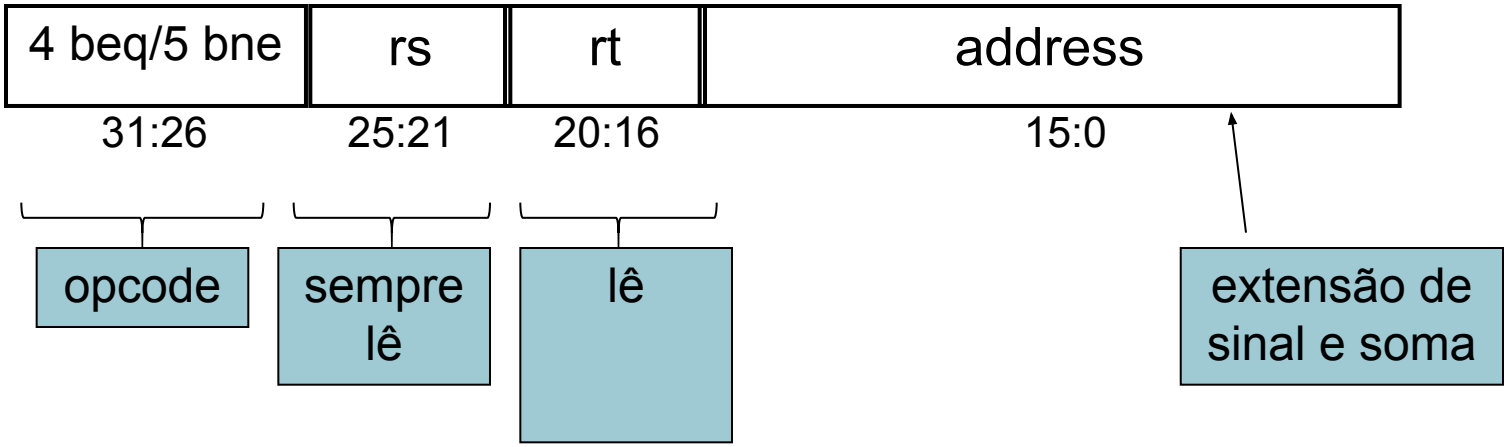
2 - Leitura de Operandos

0x0040001c bne $t0, $s5, Exit (bne $8,$21,0x00000002)																if(R[rs]!=R[rt]) PC=PC+4+BranchAddr (0x05)														
0x05						8					21																		0x0002		
0	0	0	1	0	1	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1				5				1				5				0				0				0				2		

0x0002 extendido sinal de 16 bits para 32 bits = 0x0000.0002
0x0000.0002 << 2 = 0x0000.0008 bytes

PC = (PC+4) +BranchAddr
PC = 0x00400020 + 0x00000008 = 0x00400028 (Exit: nop)

Desvio



Instruções Desvio - Leitura Reg

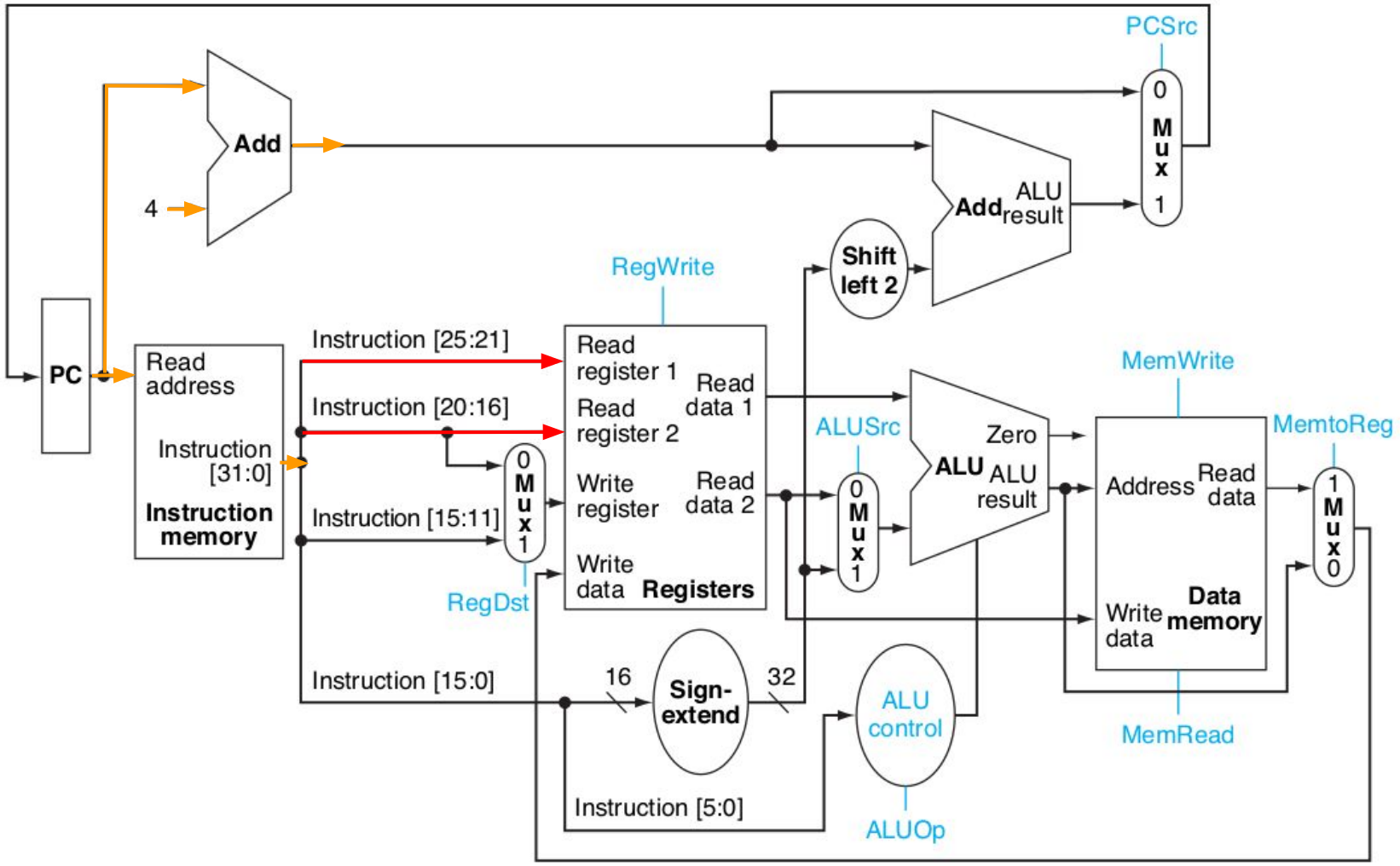


FIGURE 4.15

Instruções Desvio - Leitura Reg

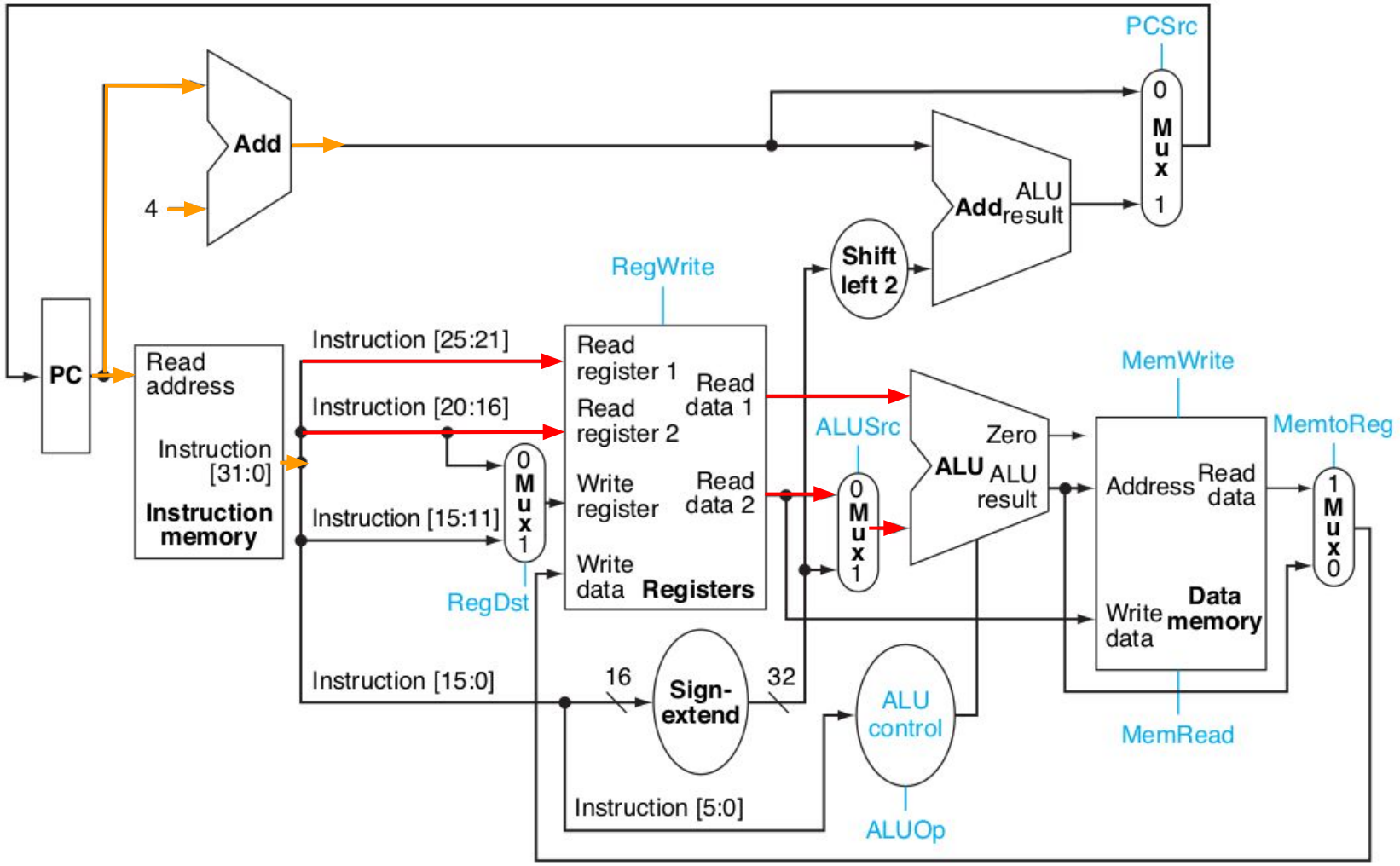


FIGURE 4.15

Instruções Desvio - Cálculo

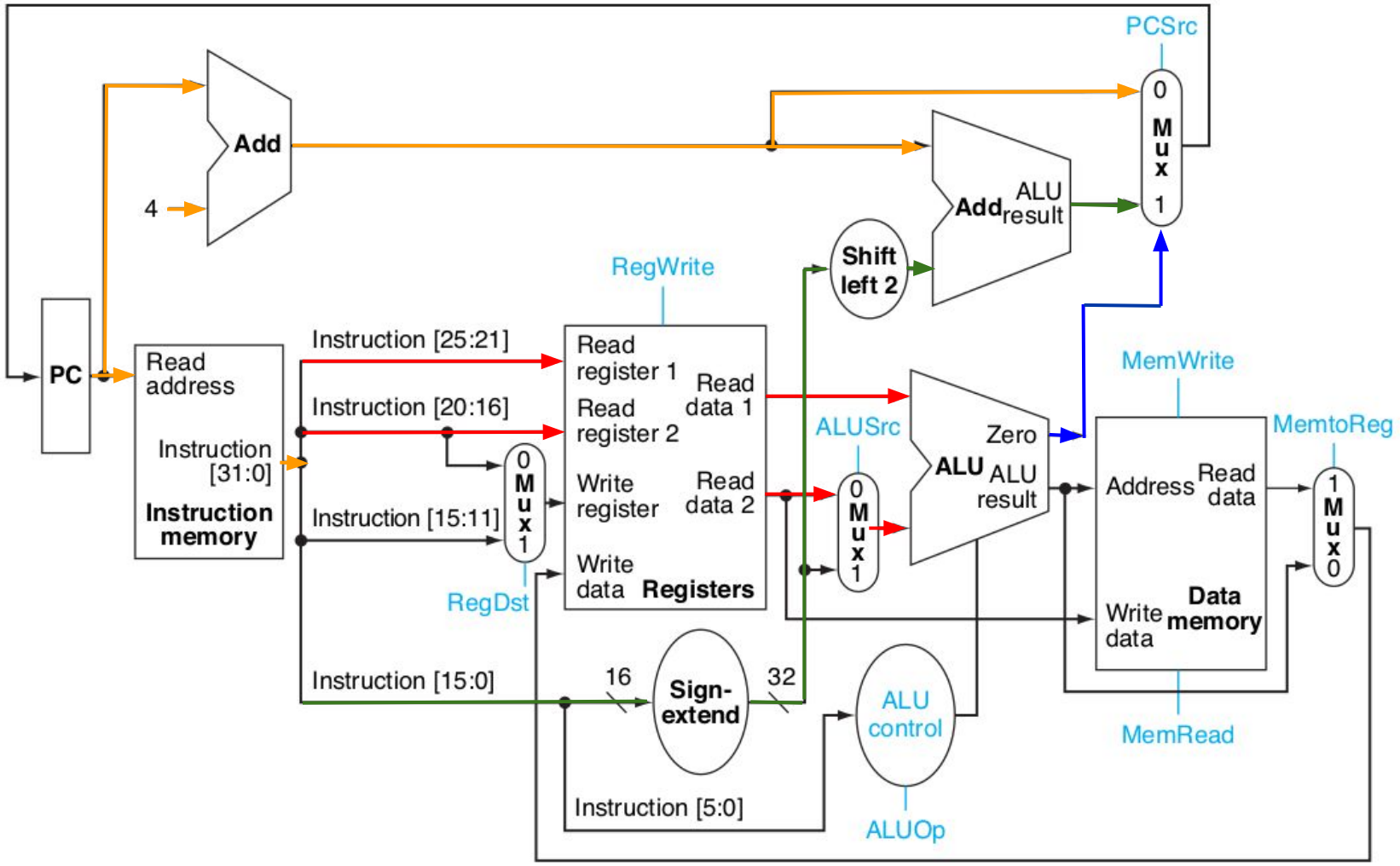


FIGURE 4.15

Instruções Desvio - Cálculo

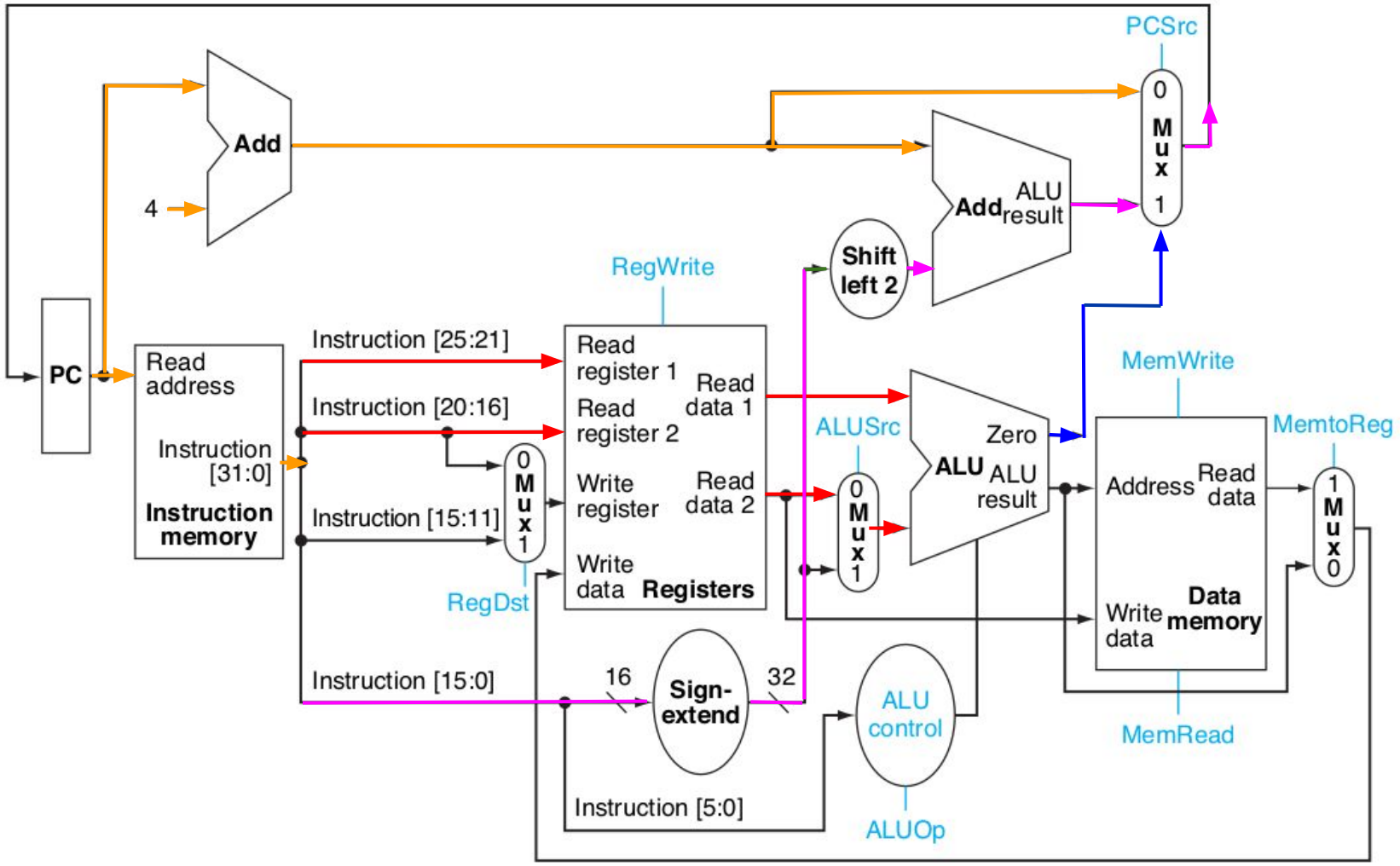


FIGURE 4.15

Instruções Desvio - Cálculo

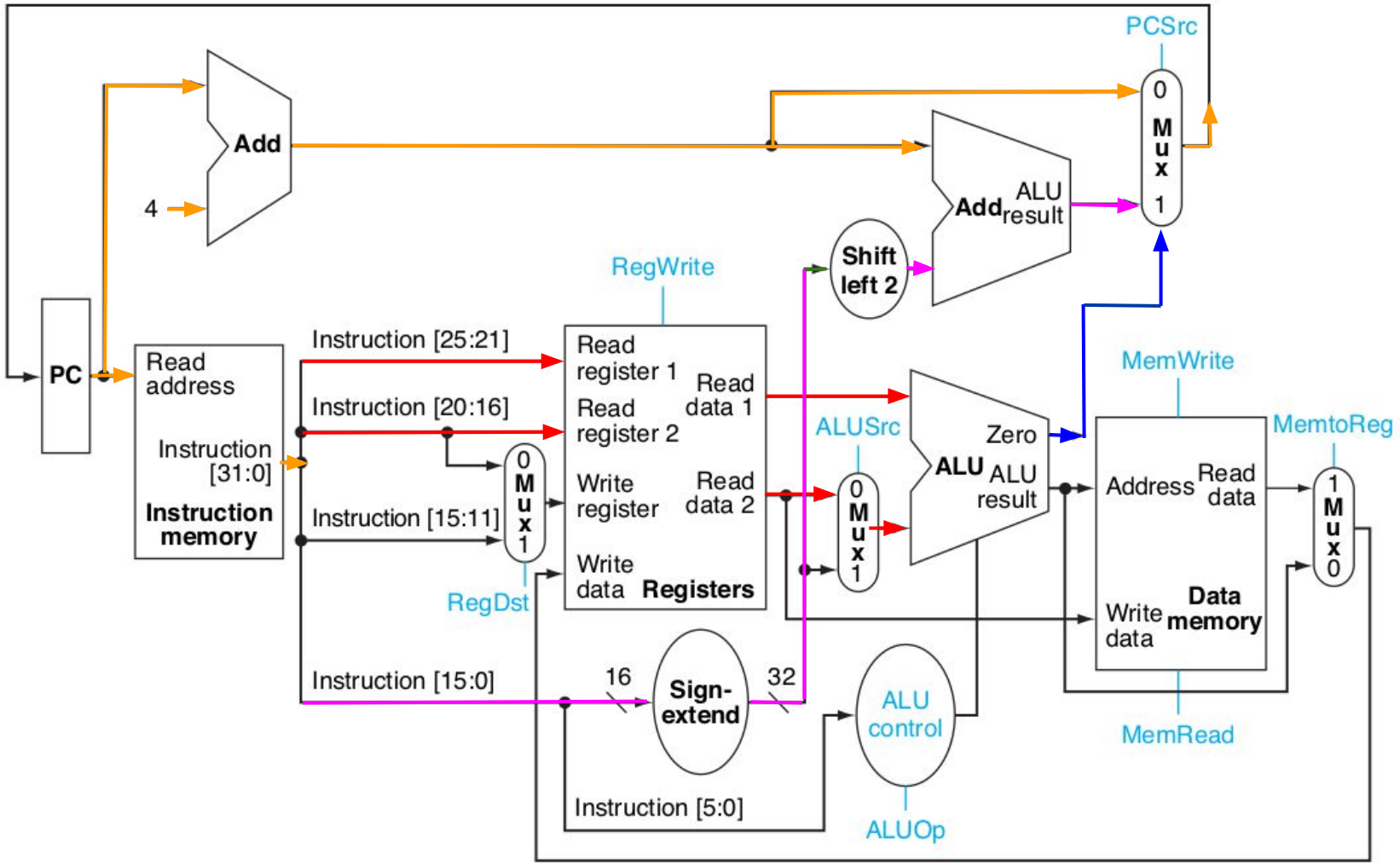


FIGURE 4.15

Compondo os Elementos

- O caminho de dados inicializado faz uma instrução em um ciclo de clock
  - Cada elemento do caminho de dados pode executar apenas uma função por vez
  - Portanto, precisamos de instruções separadas e memórias de dados
- Usar multiplexadores onde fontes de dados alternativas são usadas para instruções diferentes

Datapath Completo

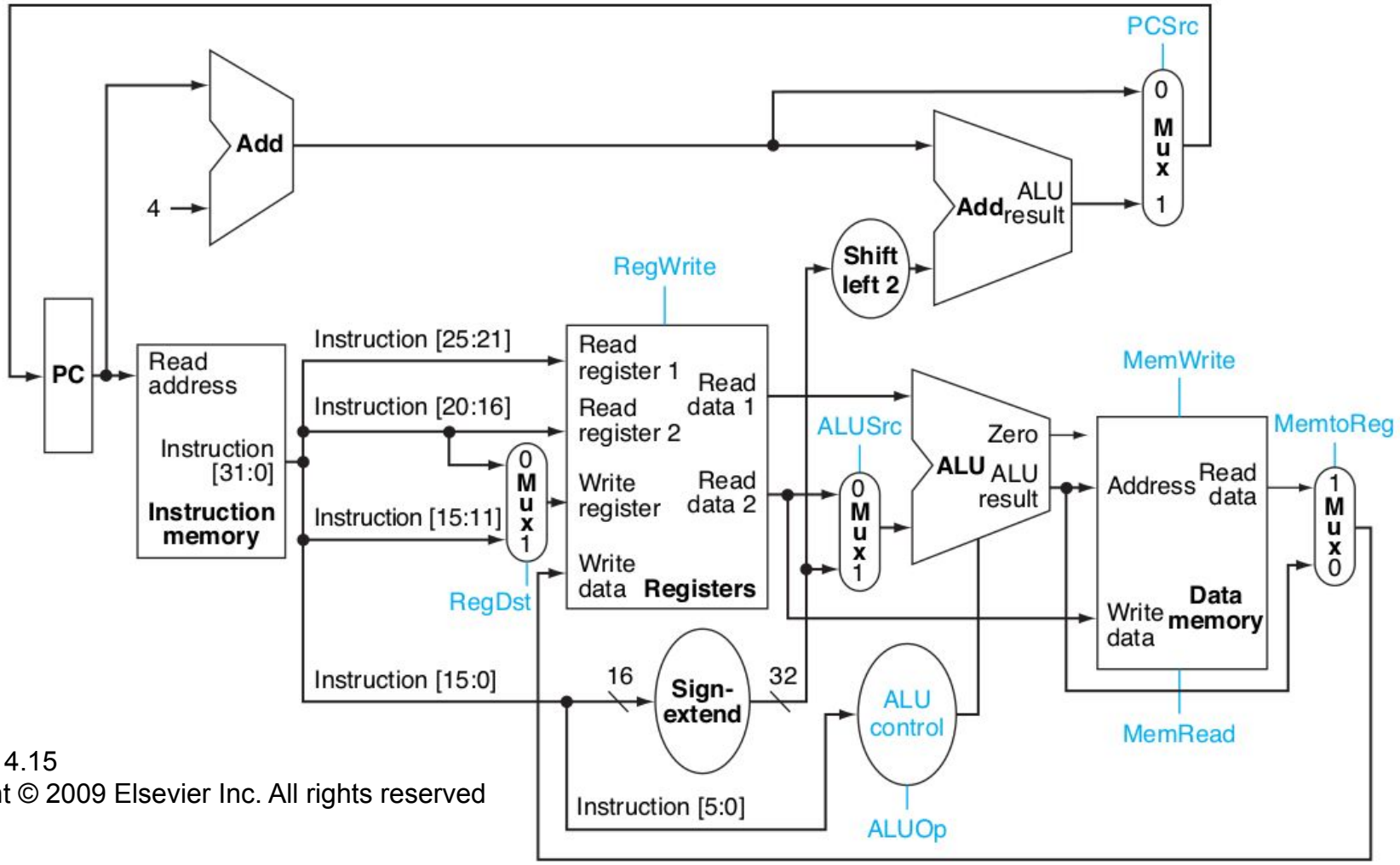


FIGURE 4.15

Referências

- Seções 4.1 a 4.3 - “Organização e Projeto de Computadores - A Interface Hardware/Software, David A. Patterson & John L. Hennessy, Campus, 4 edição, 2013.