Arquitetura e Organização de Computadores

Cap 4. O Processador

Parte 5 - Datapath monociclo

Prof. Dr. João Fabrício Filho

Universidade Tecnológica Federal do Paraná
Campus Campo Mourão
2024

Introdução

Fatores de desempenho da CPU

- Contagem de instruções
  - Determinada pela ISA e pelo compilador
- CPI e período de clock
  - Determinado pelo hardware da CPU

Examinaremos duas implementações MIPS

- Versão simplificada
- Uma versão mais realística com implementação de pipeline

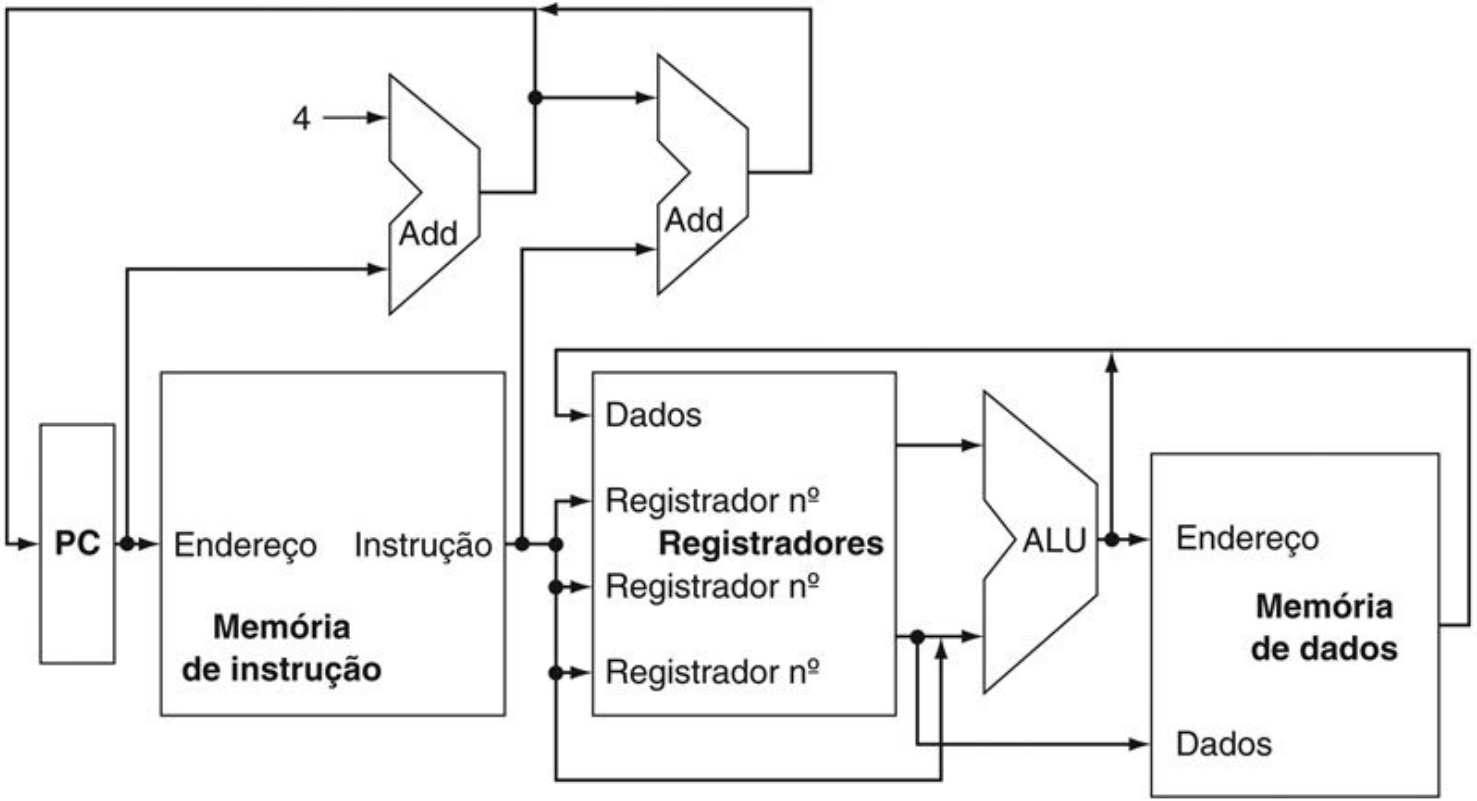
Subconjunto simples, apresenta a maioria dos aspectos

- Referência à memória: lw, sw
- Lógica/Aritmética: add, sub, and, or, slt
- Transferência de controle: beq, j

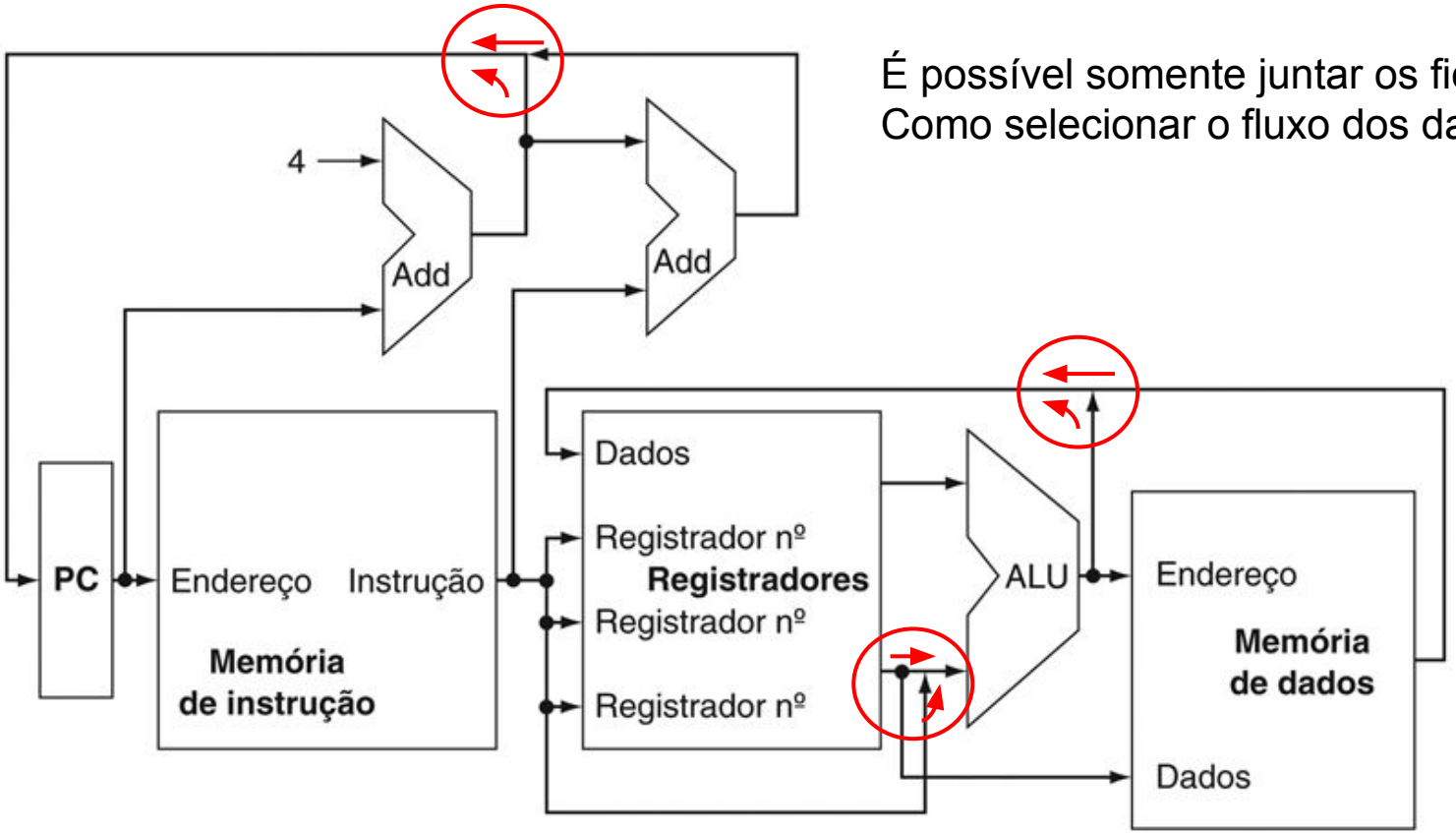
Execução de instrução

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

Visão geral da CPU

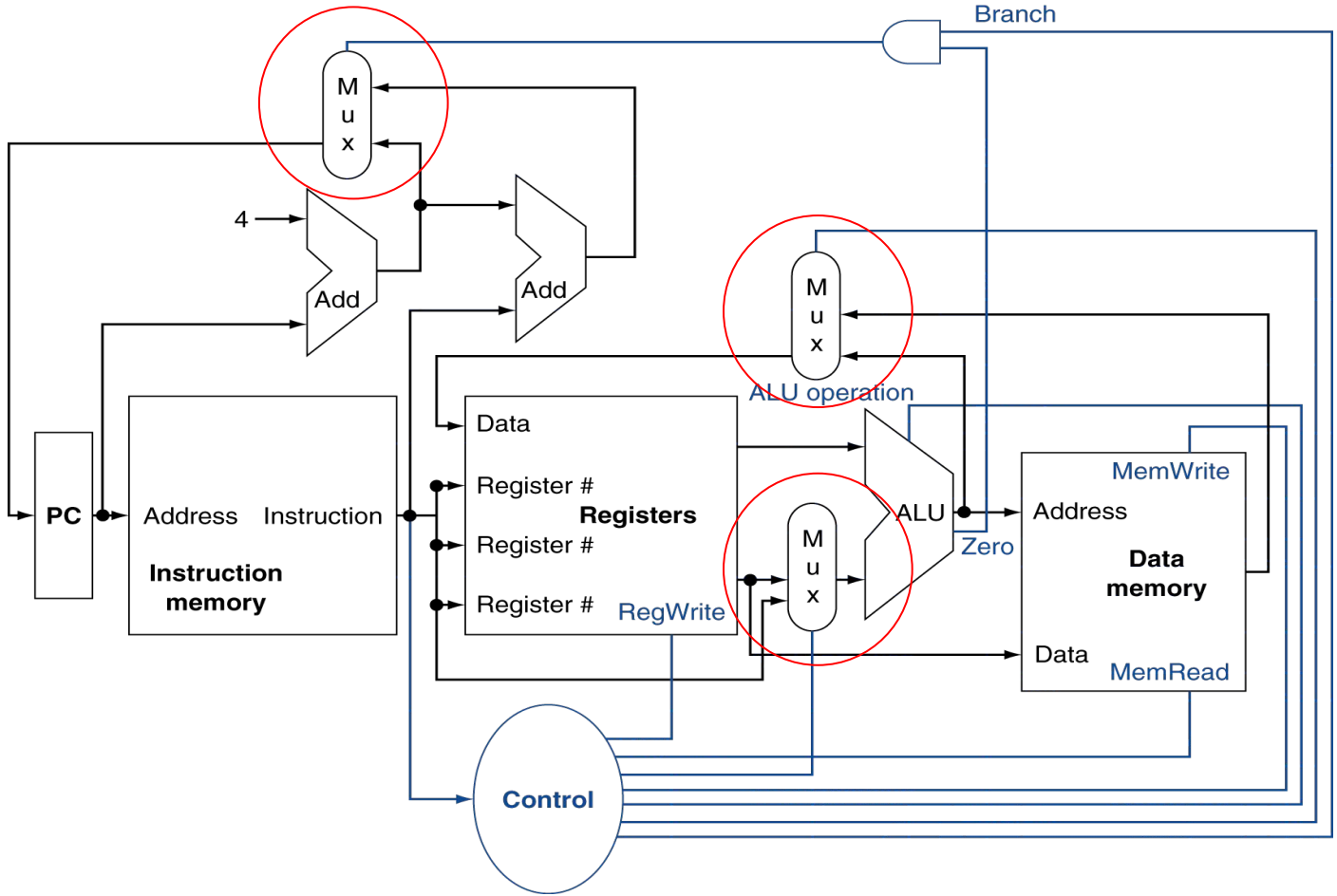


Visão geral da CPU

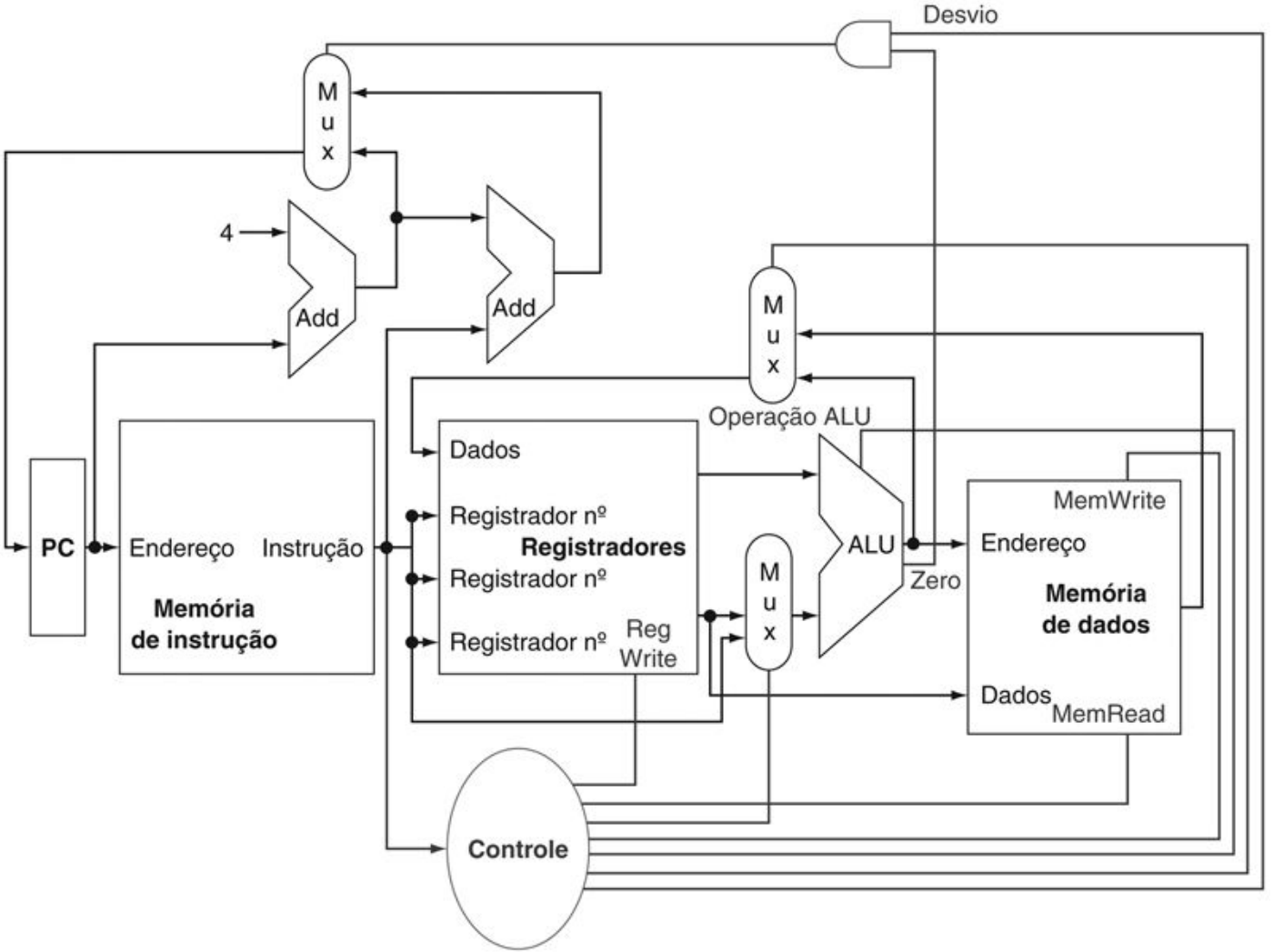


É possível somente juntar os fios?
Como selecionar o fluxo dos dados?

Multiplexadores



Datapath


Desvio
M
u
x
4
Add
Add
M
u
x
Operação ALU
PC
Endereço
Instrução
Memória
de instrução
Dados
Registrador nº
Registradores
Registrador nº
Registrador nº
Reg
Write
M
u
x
ALU
Zero
MemWrite
Endereço
Memória
de dados
Dados
MemRead
Controle

Como uma instrução pode transformar o estado da computação?

O que é estado?

Quais componentes definem o estado?

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

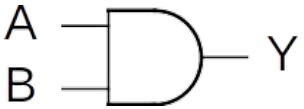
- Opera sobre os dados
- Saída é função das entradas

Elemento de estado (sequenciais)

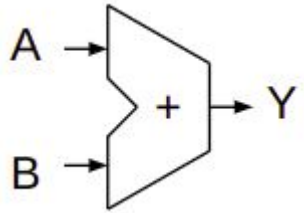
- Armazena informação

Elementos combinacionais

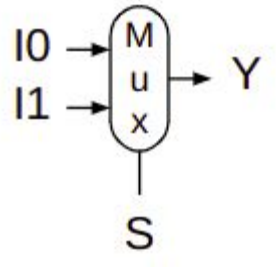
- Porta AND
  - Y = A & B



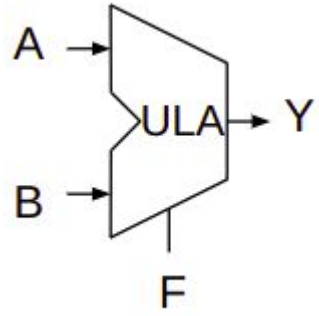
- Somador
  - Y = A + B



- Multiplexador
  - Y = S? I0 : I1



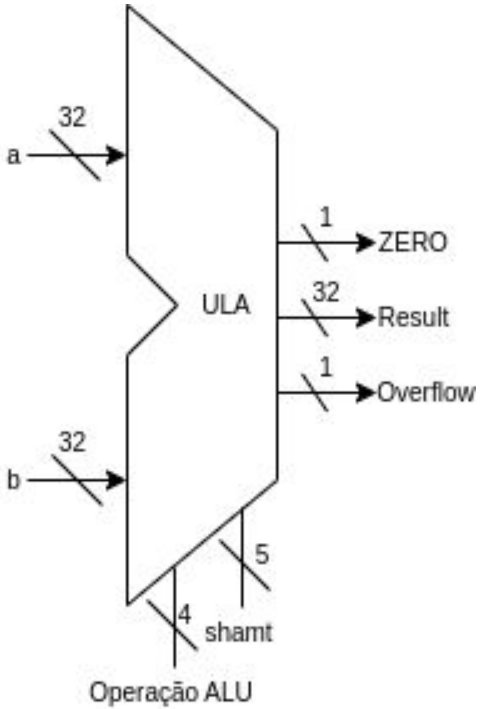
- ULA
  - Y = F(A, B)



Unidade Lógica e Aritmética

Realiza uma operação entre a e b

- A operação depende do que está especificado em Operação ALU
- Result contém o resultado da operação

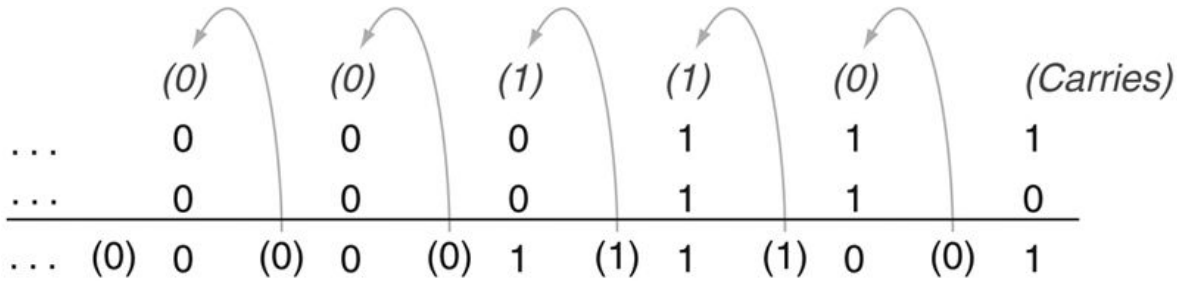


Qual a saída do código ao lado?

```c
int a = 2147483647;
a++;
printf("%d", a);
```

Adição de inteiros

- Exemplo: 7+6



- Overflow se o resultado estiver fora da faixa permitida
  - Soma com + e −, sem overflow
  - Soma dois positivos
    - Overflow se o sinal do resultado for 1
  - Soma dois negativos
    - Overflow se o sinal do resultado for 0

Subtração de inteiros

- Negação do segundo operando
- Exemplo: 7 - 6 = 7 + (-6)
  - +7: 0000 0000 … 0000 0111
  - -6: 1111 1111 … 1111 1010
  - +1: 0000 0000 … 0000 0001
- Overflow se o resultado estiver fora da faixa
  - Subtração de dois + ou dois -, sem overflow
  - Subtrair + de um valor -
    - Overflow se o sinal for 0
  - Subtrair - de um valor +
    - Overflow se o sinal for 1

Condições de overflow

Operação	Operando A	Operando B	Resultado indicando overflow
$A + B$	≥ 0	≥ 0	< 0
$A + B$	< 0	< 0	≥ 0
$A - B$	≥ 0	< 0	< 0
$A - B$	< 0	≥ 0	≥ 0

Unidade lógica aritmética

- **ALU de 1 bit - Figura 4.8 – blocos usados para construção de uma ALU**

1. AND gate (c = a . b)

a	b	c = a . b
0	0	0
0	1	0
1	0	0
1	1	1

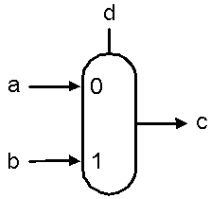
2. OR gate (c = a + b)

a	b	c = a + b
0	0	0
0	1	1
1	0	1
1	1	1

3. Inverter (c = a)

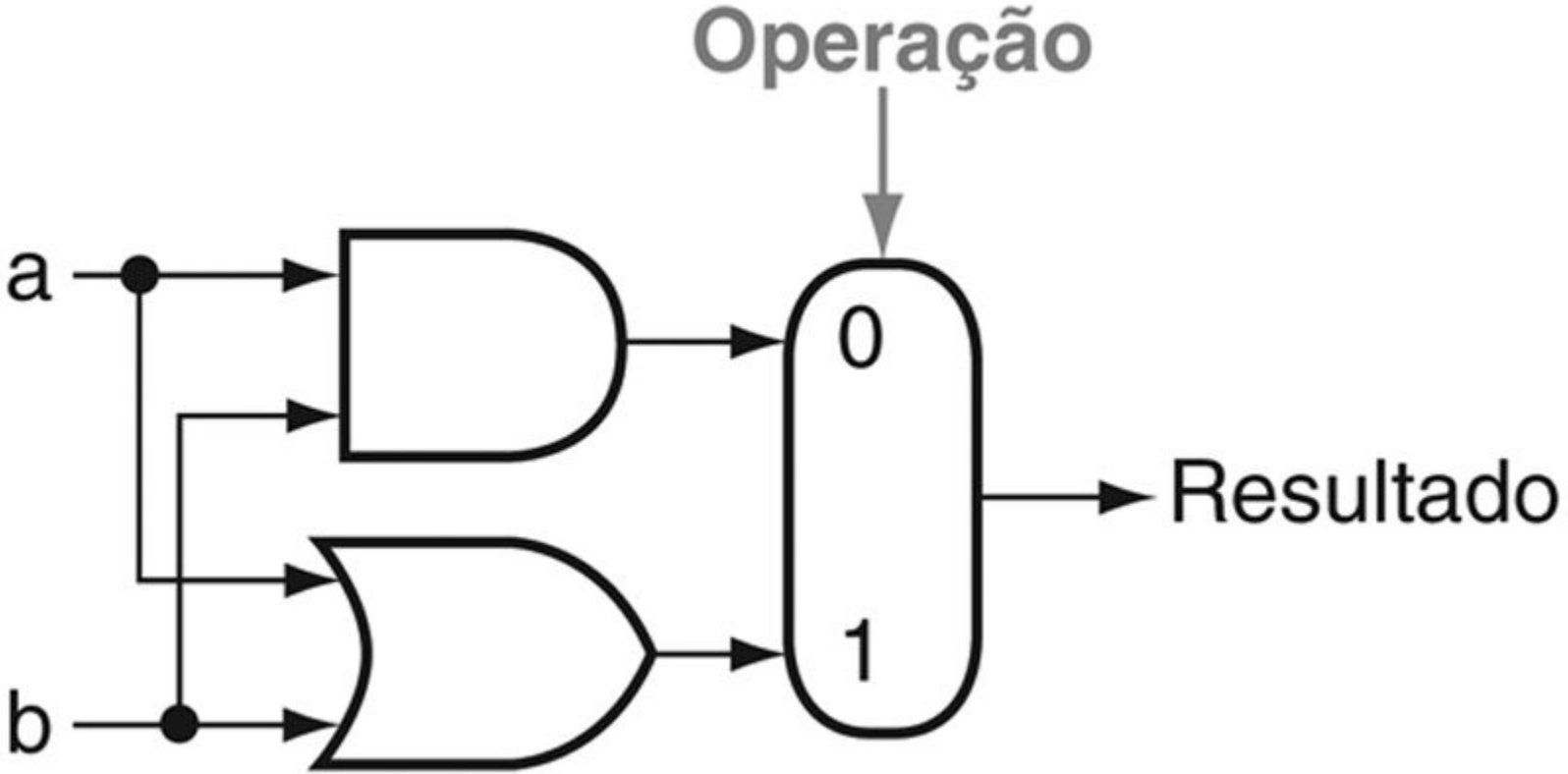
a	c = $\bar{a}$
0	1
1	0

4. Multiplexor
(if d = = 0, c = a;
else c = b)

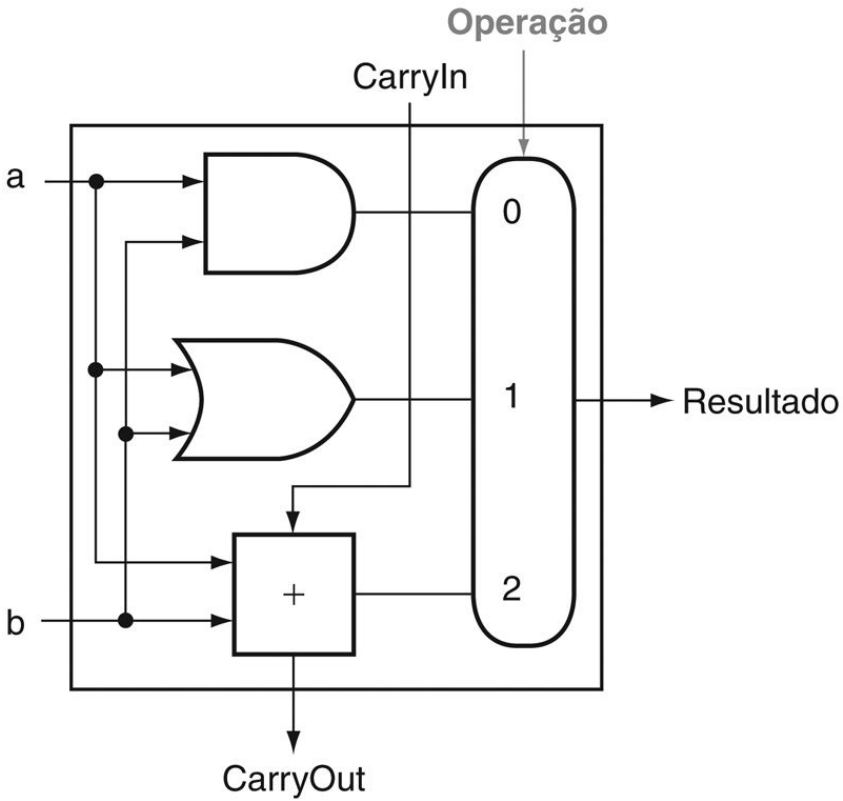


d	c
0	a
1	b

ULA de 1 bit para AND/OR



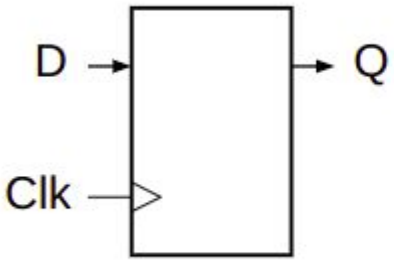
ULA de 1 bit para AND/OR/ADD

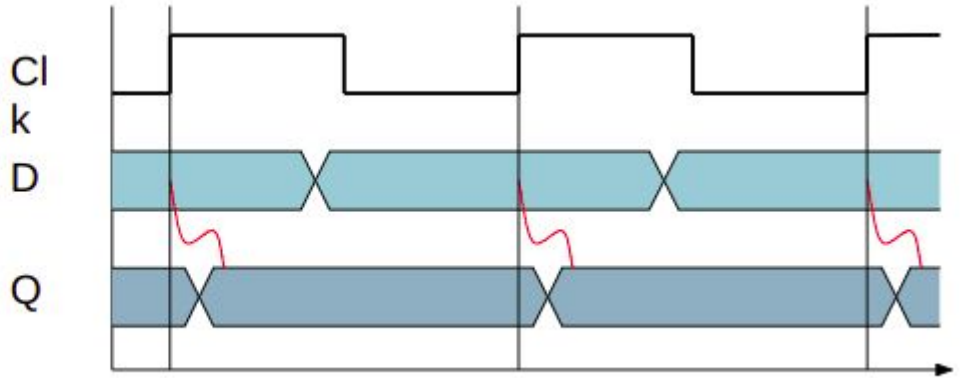


Elementos de estado

Registrador: armazena dados em um circuito

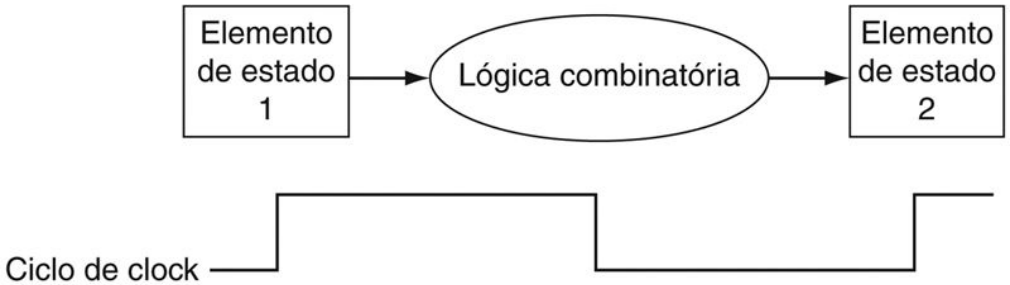
- Usa o sinal de clock para determinar quando o valor armazenado será atualizado
- Acionado por borda: atualiza quando o clock muda de 0 para 1





Funcionamento do clock

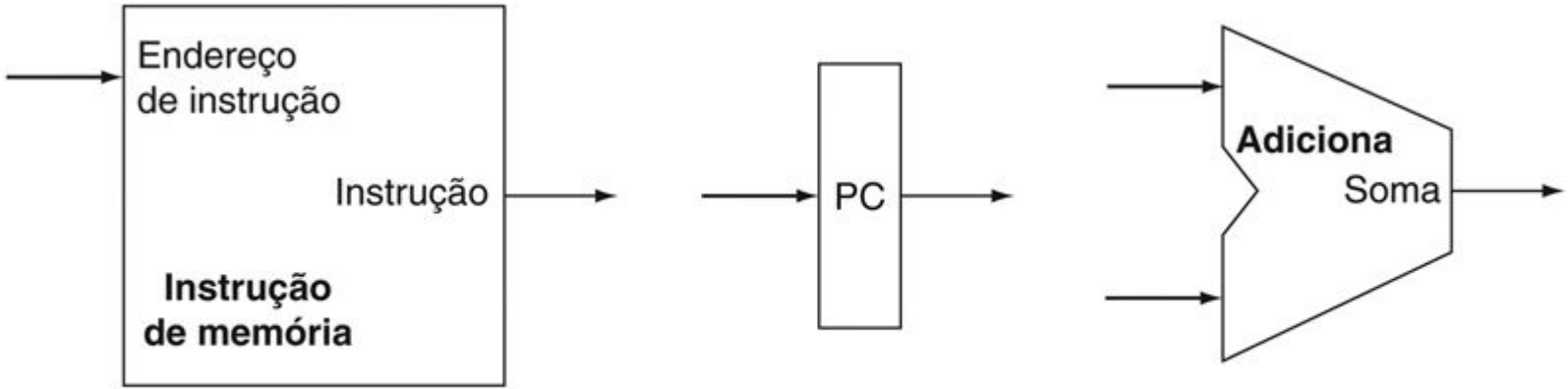
- A lógica combinacional muda os dados durante os ciclos de clock
  - Entre as bordas do clock
  - Entrada de elementos de estado, saída para elementos de estado
  - Atrasos longos nas instruções determinam o período do clock



Construindo um datapath

- Datapath
  - Elementos que processam dados e endereços na CPU
    - Registradores, ALUs, Muxes, Memórias
- Construiremos um datapath MIPS de maneira incremental
  - Refinando o projeto

Elementos da busca de instrução



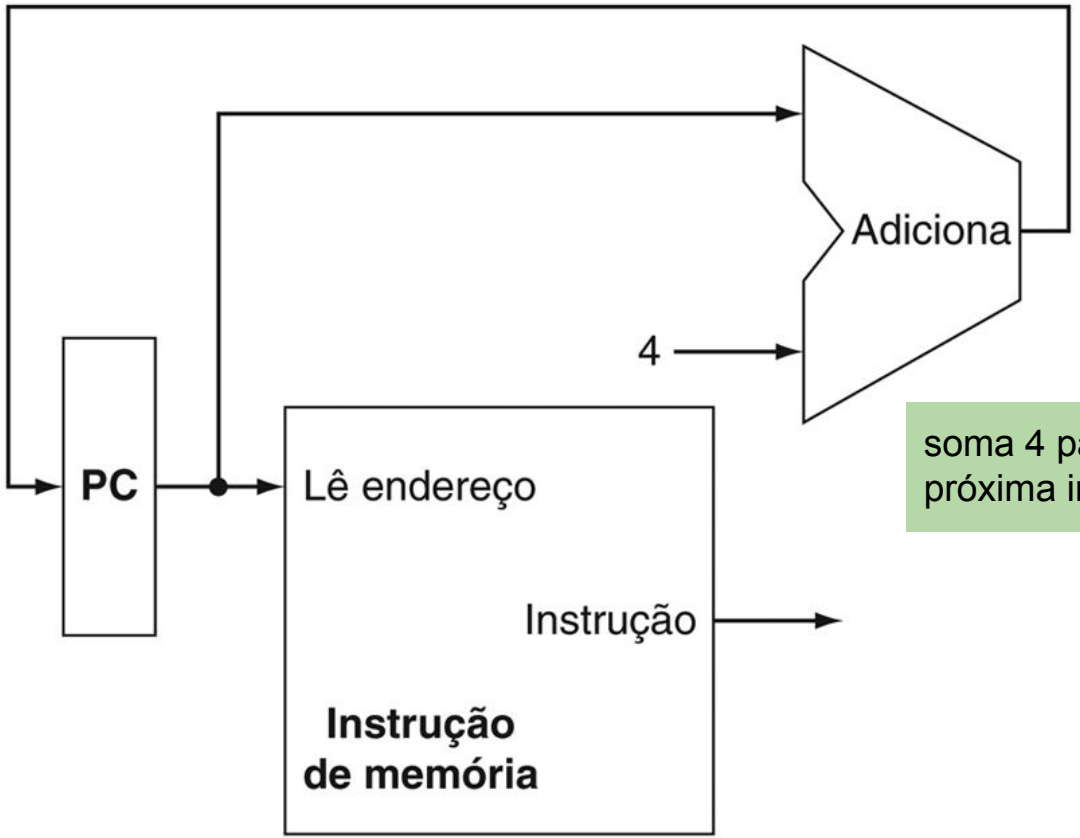
a. Instrução de memória

b. Contador de programa

c. Adicionador

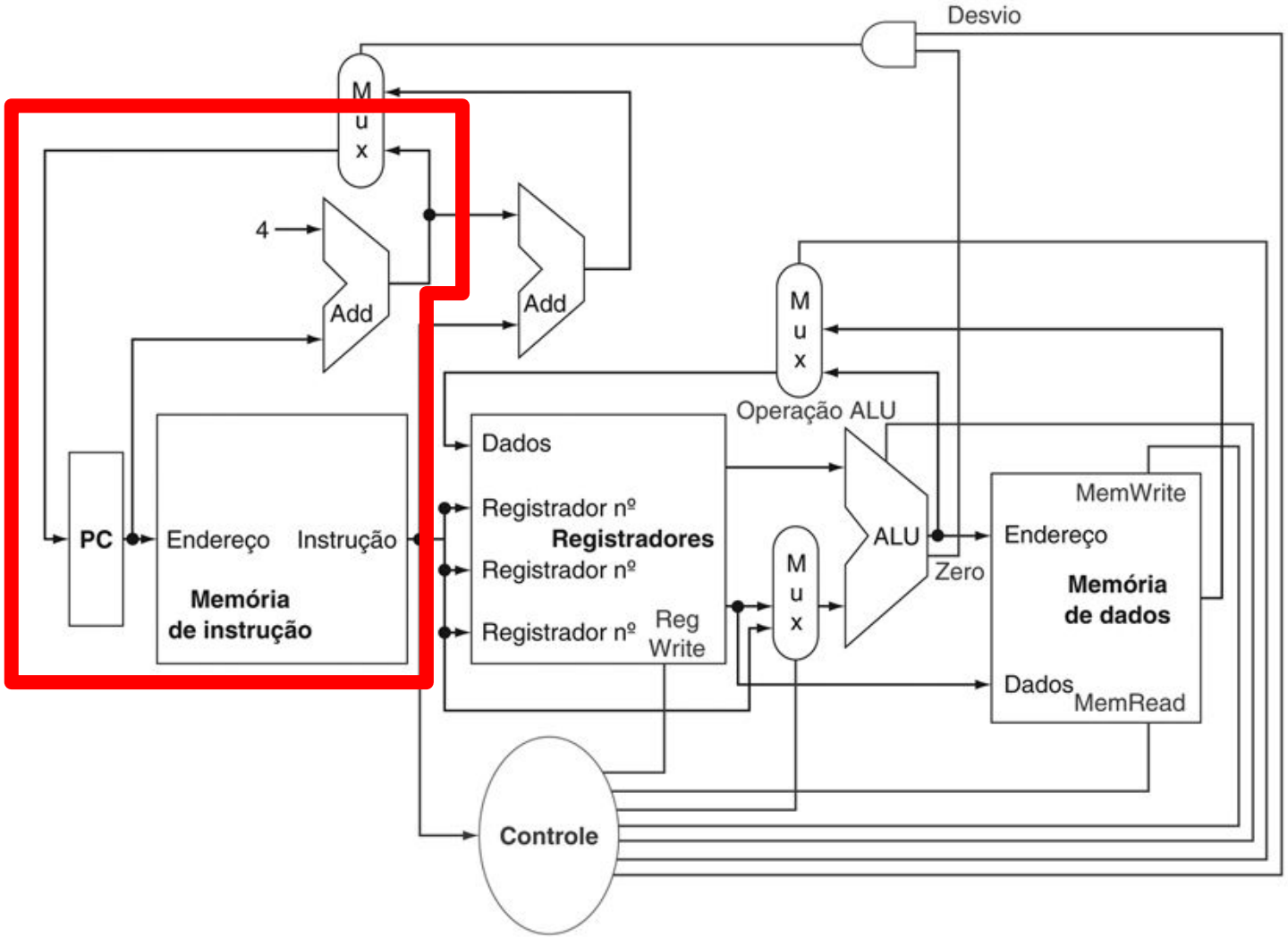
registrador de 32 bits

Busca de instrução

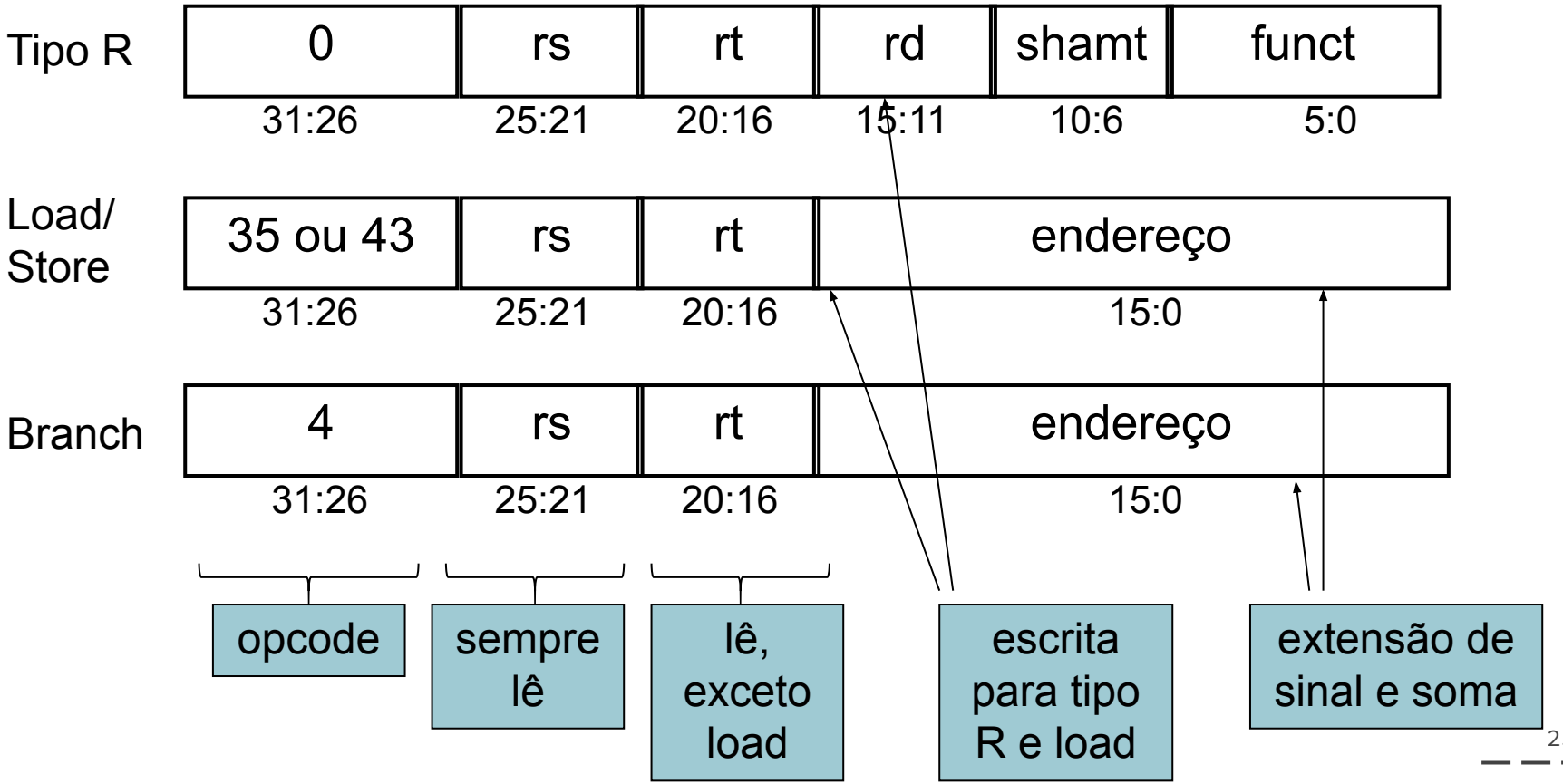


soma 4 para buscar a próxima instrução

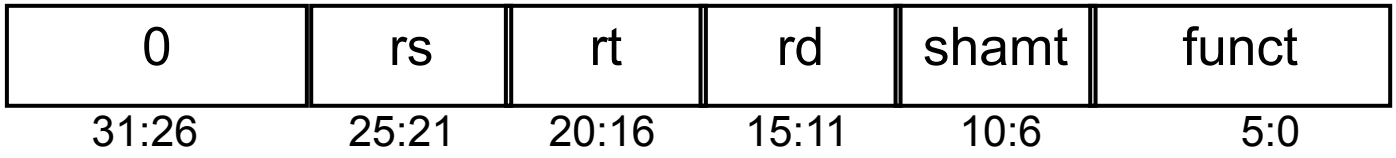
Datapath


Desvio
M
u
x
4
Add
Add
M
u
x
Operação ALU
PC
Endereço
Instrução
Memória
de instrução
Dados
Registrador nº
Registradores
Registrador nº
Registrador nº
Reg
Write
M
u
x
ALU
Zero
MemWrite
Endereço
Memória
de dados
Dados
MemRead
Controle

Leitura de operandos



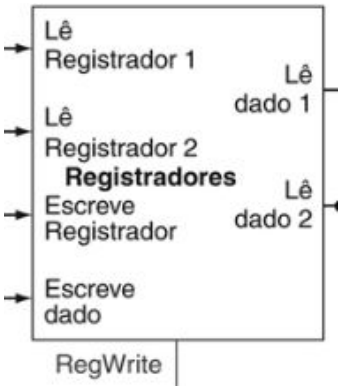
Instrução do tipo R

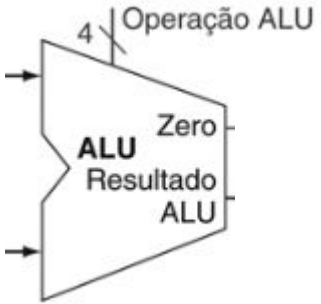


Lê dois operandos dos registradores rs e rt

Executa operação da ULA

Escreve o resultado no registrador rd





Execução de instrução formato R

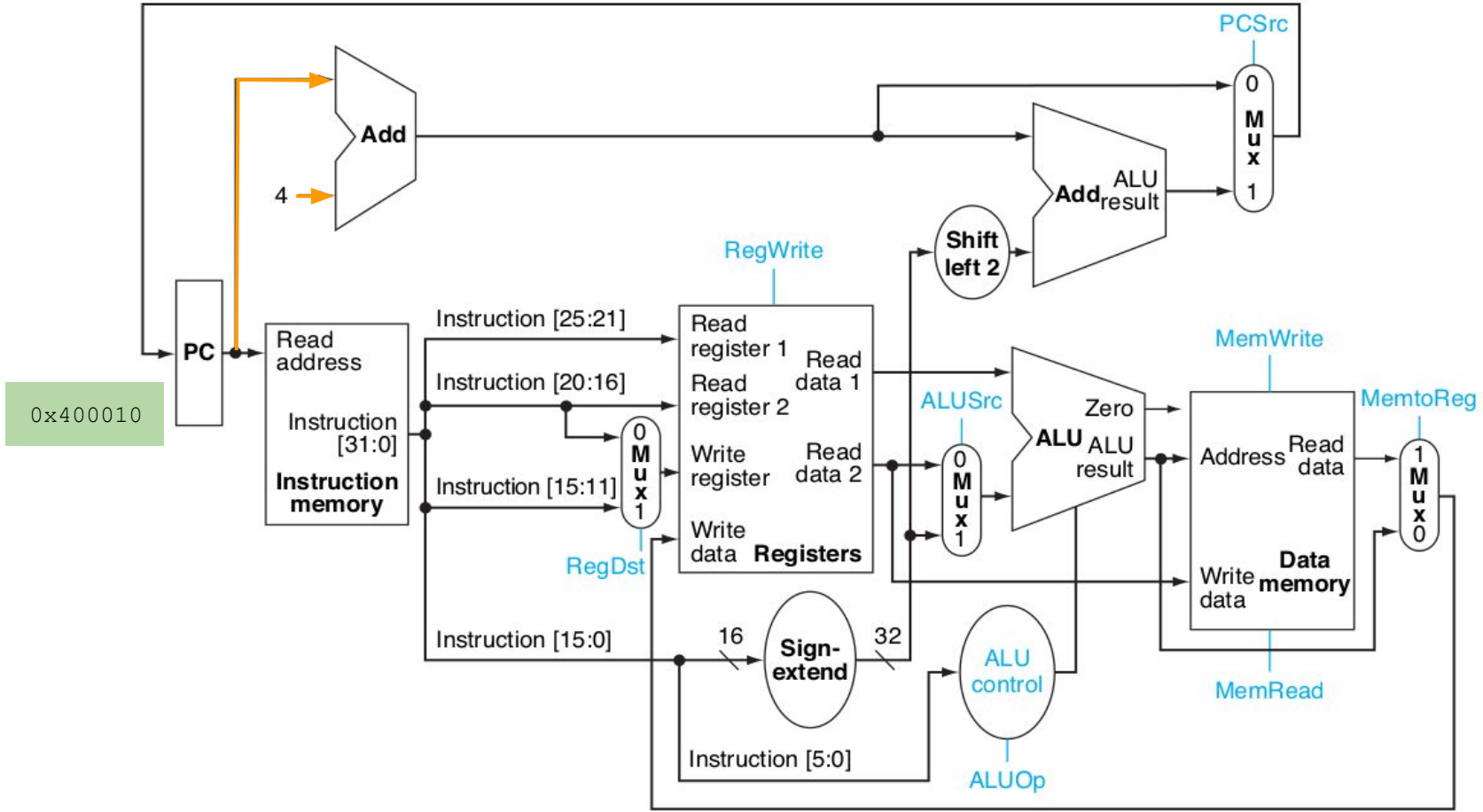
0x00400014 add $t1,$t1,$s6 (add $9,$9,$22)																R[rd] = R[rs] + R[rt] (0/20 hex)															
0x00						9					22					9									(0x20)							
0	0	0	0	0	0	0	1	0	0	1	1	0	1	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0
0				1				3				6				4				8				2				0			

Tipo R

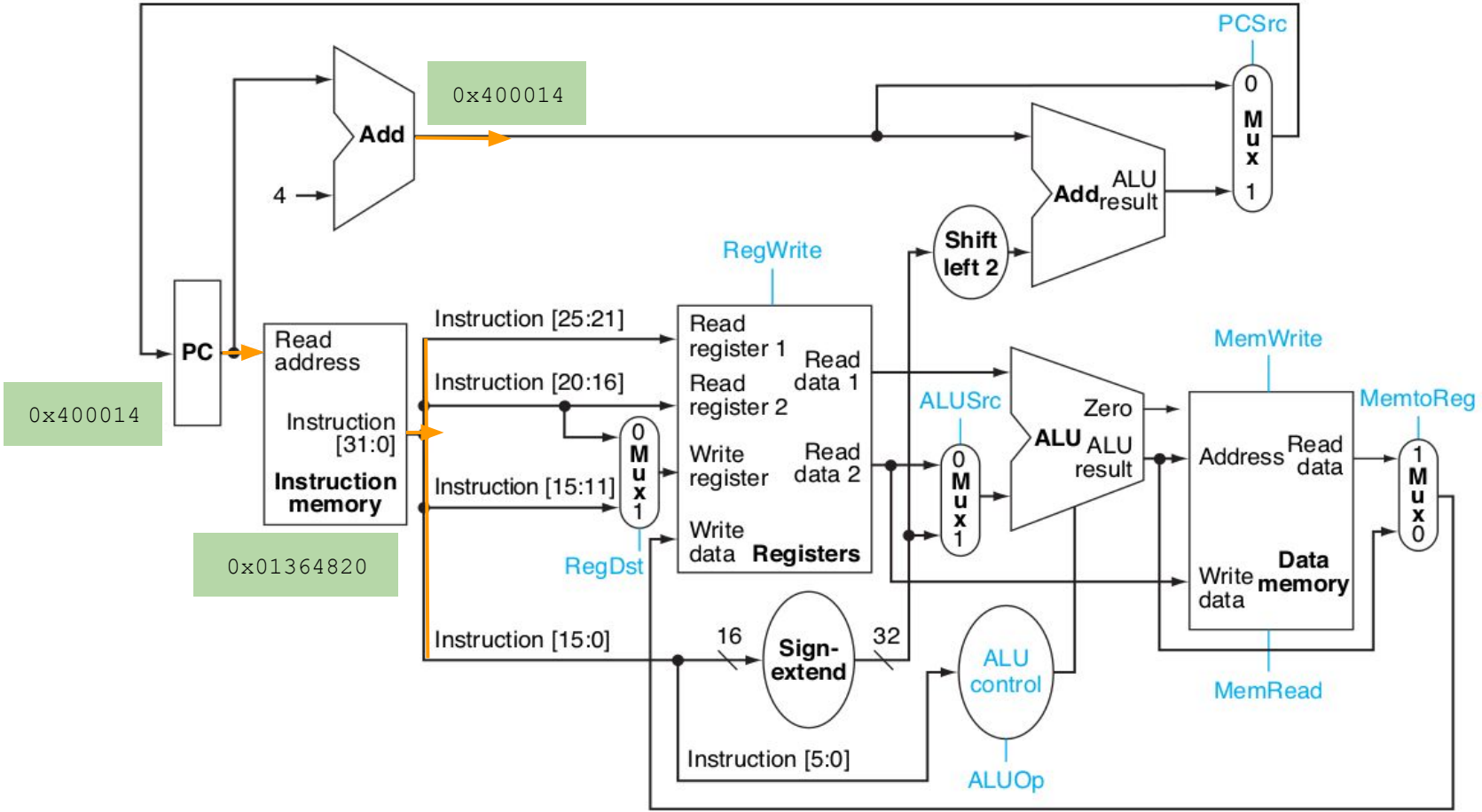
opcode	rs	rt	rd	shamt	funct
0	9	22	9	0	0x20

mnemônico: add $t1, $t1, $s6
endereço da instrução: 0x00400014
hex: 0x01364820

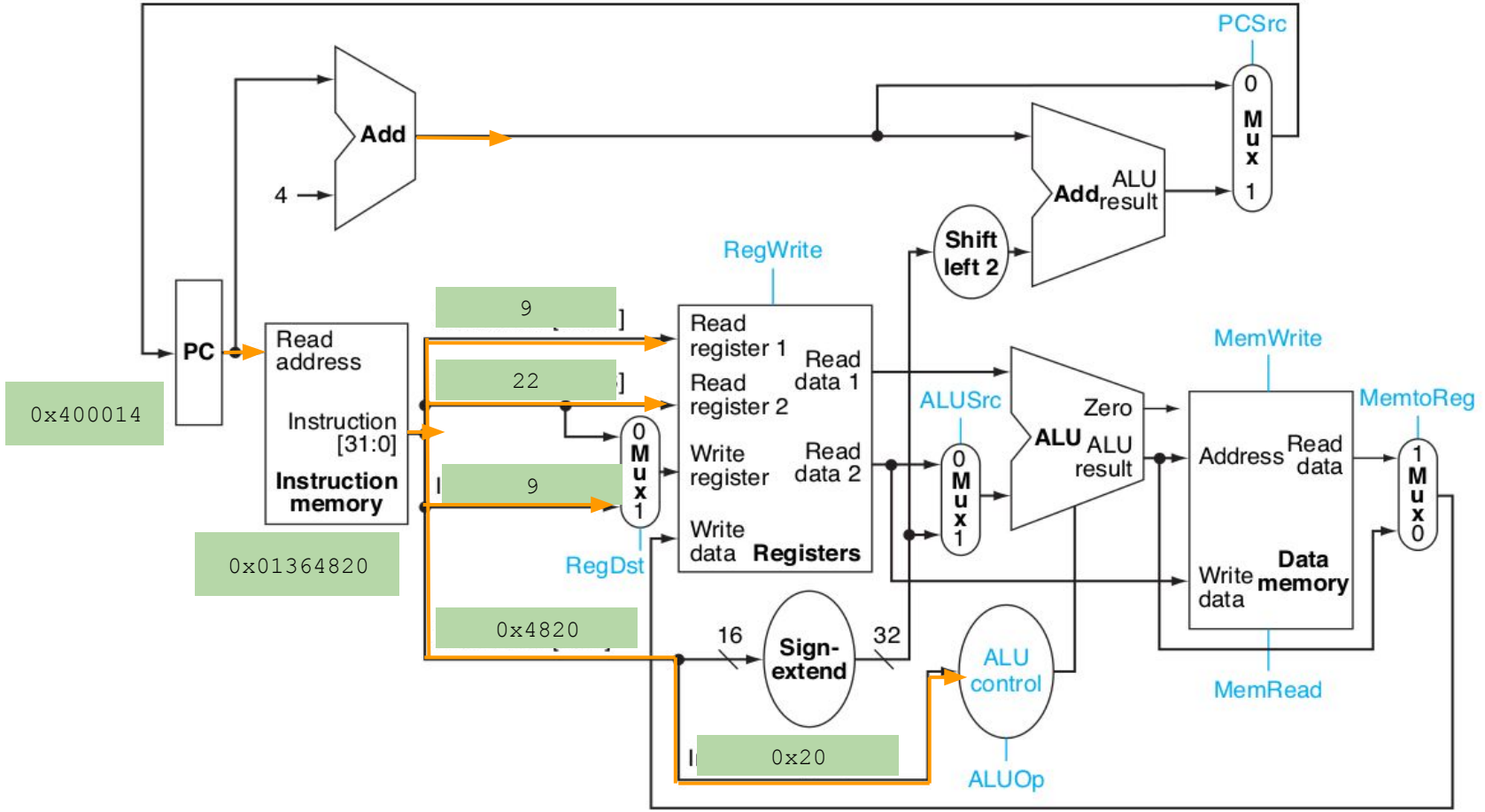
Antes da busca, PC está no endereço 0x400010



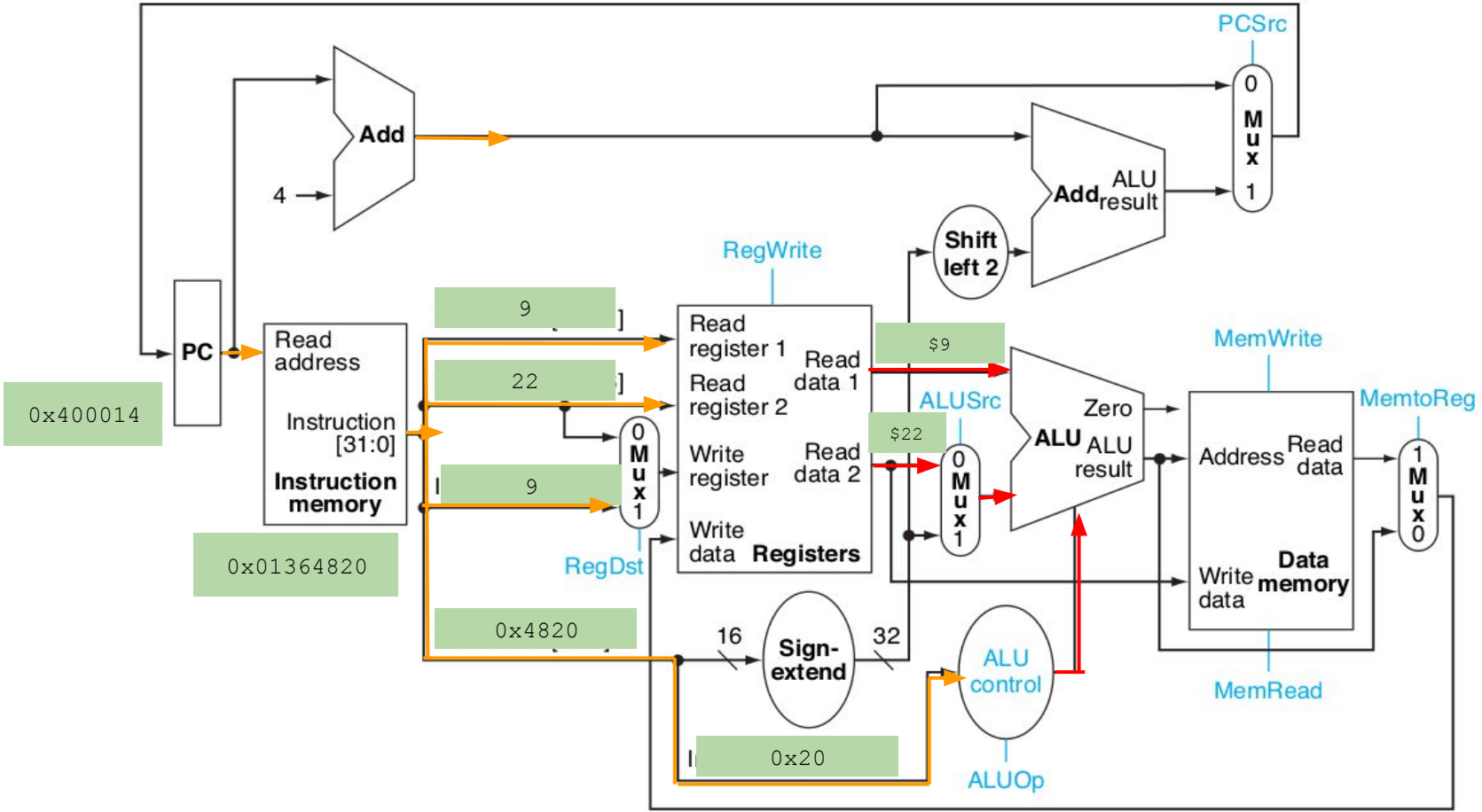
PC é incrementado em 4, uma nova instrução é buscada



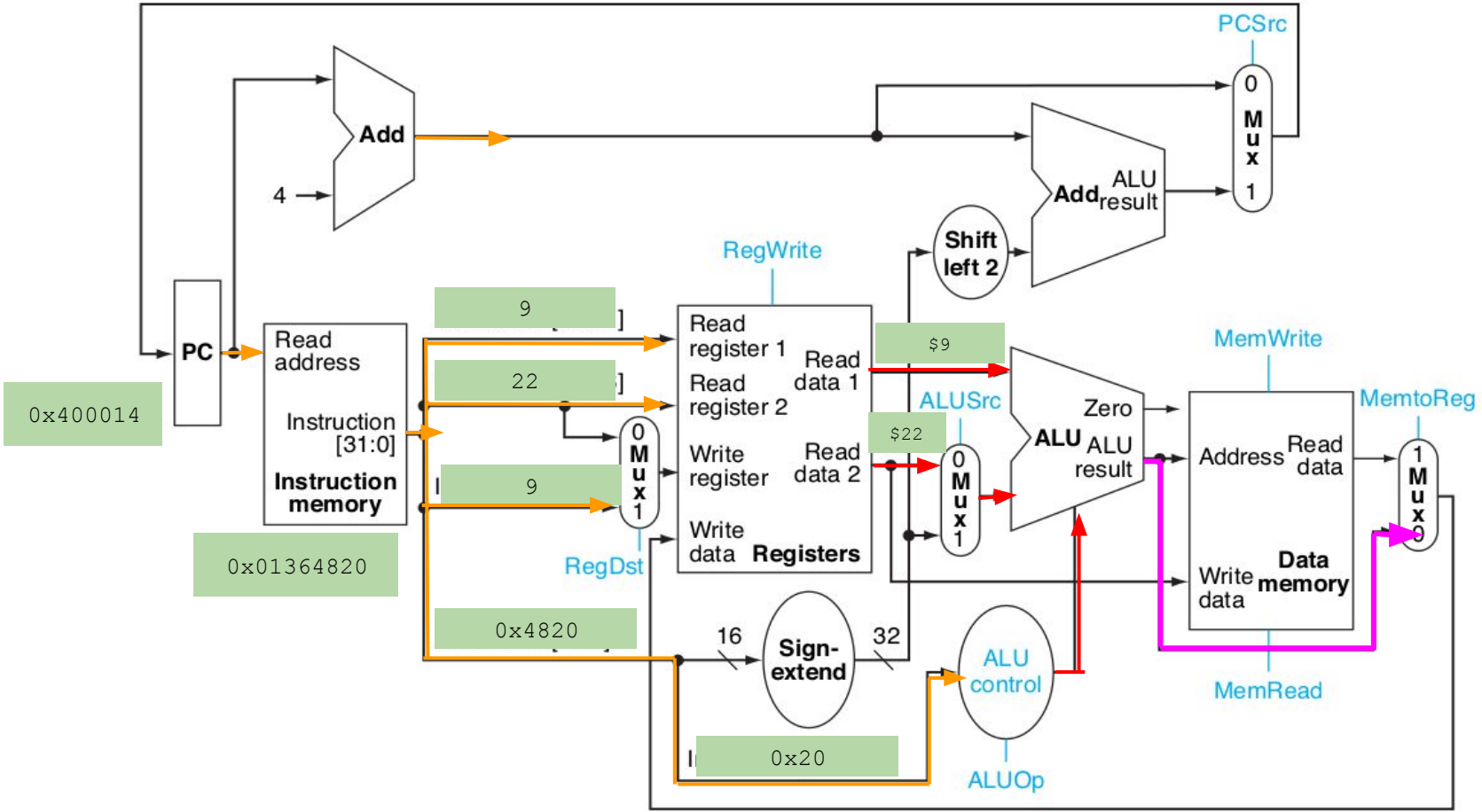
A instrução é decodificada, os sinais de controle são enviados



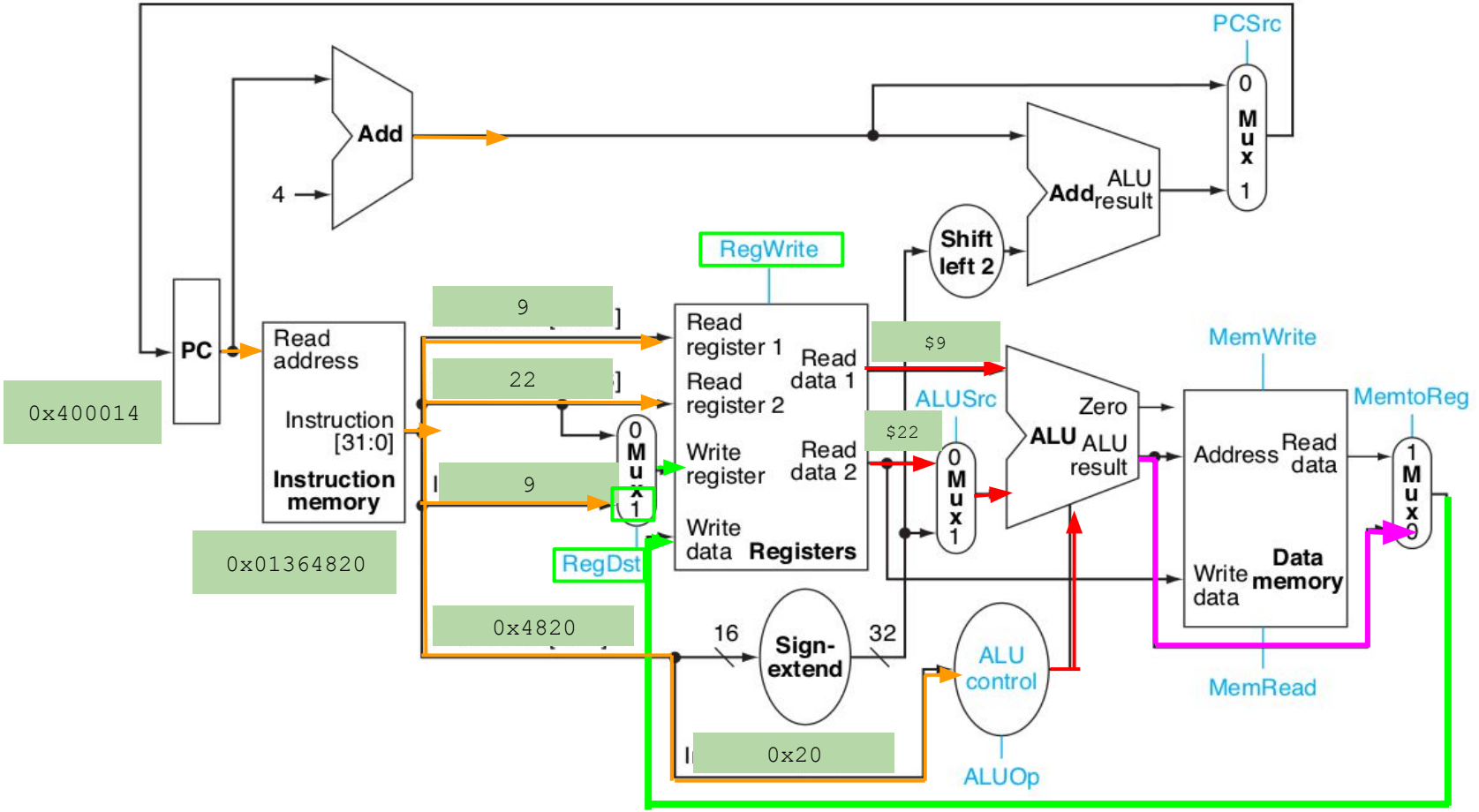
Os dados são lidos, e a operação selecionada



A operação é executada



Os dados de saída são escritos no registrador destino

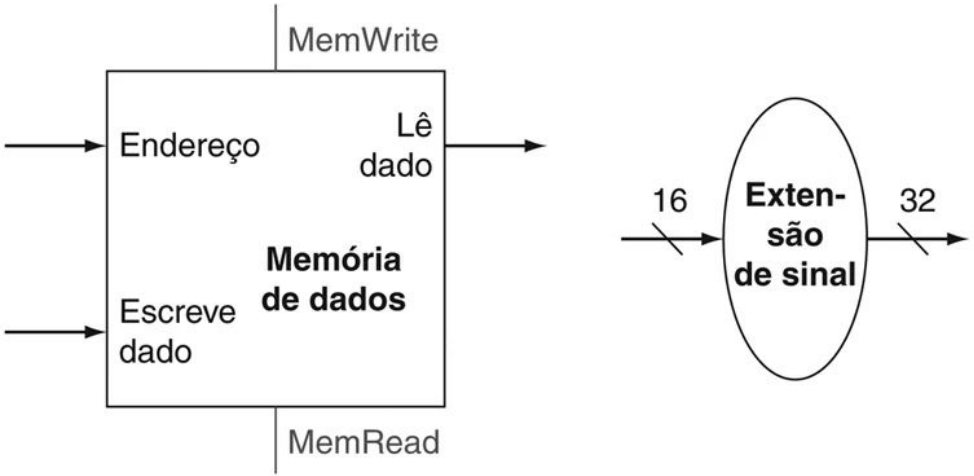


Instruções load/store

Lê operandos dos registradores
Calcula 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. Unidade de memória de dados b. Unidade de extensão de sinal

Exemplo de instrução load

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	

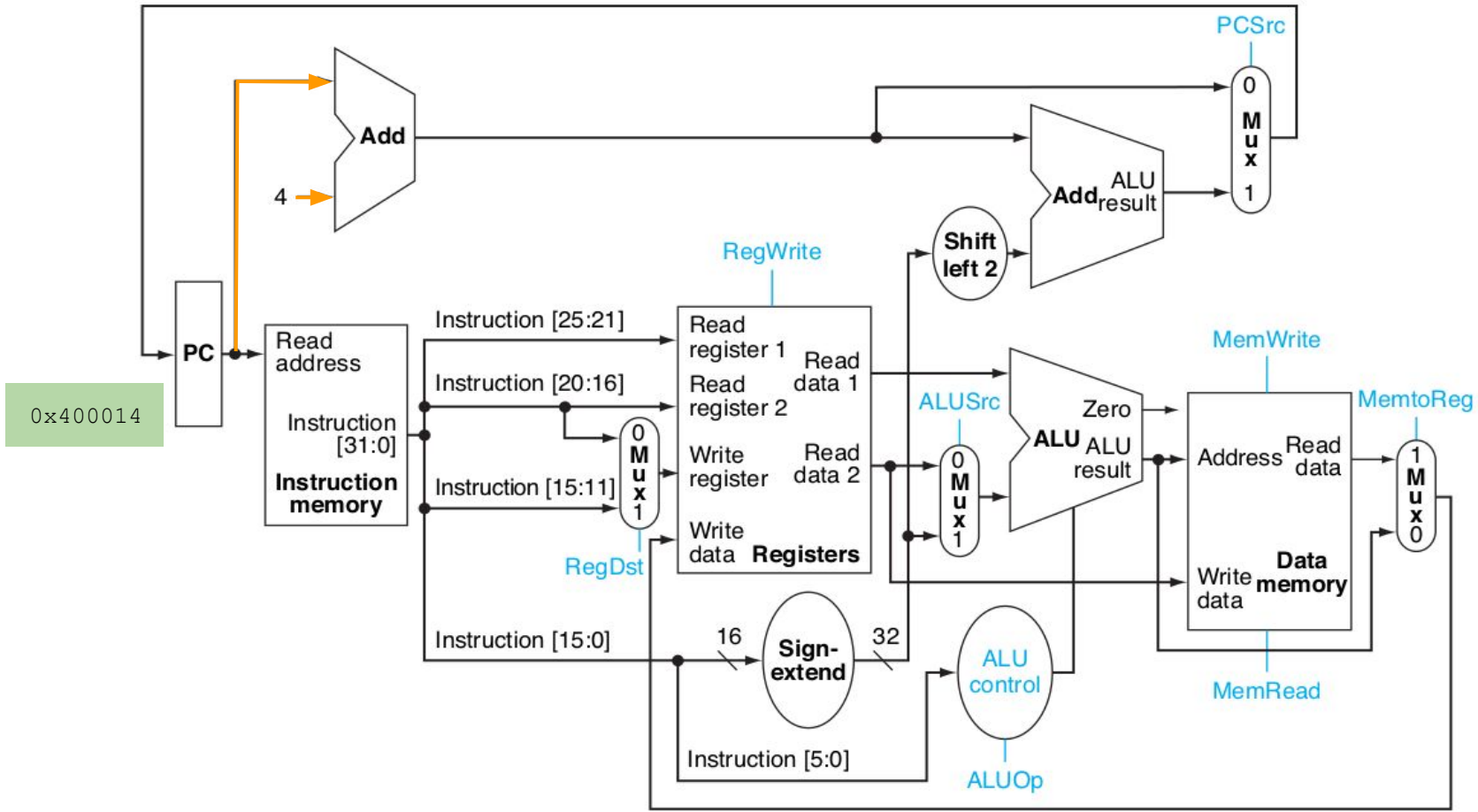
Tipo I

opcode	rs	rt	endereço
35	9	8	0x0

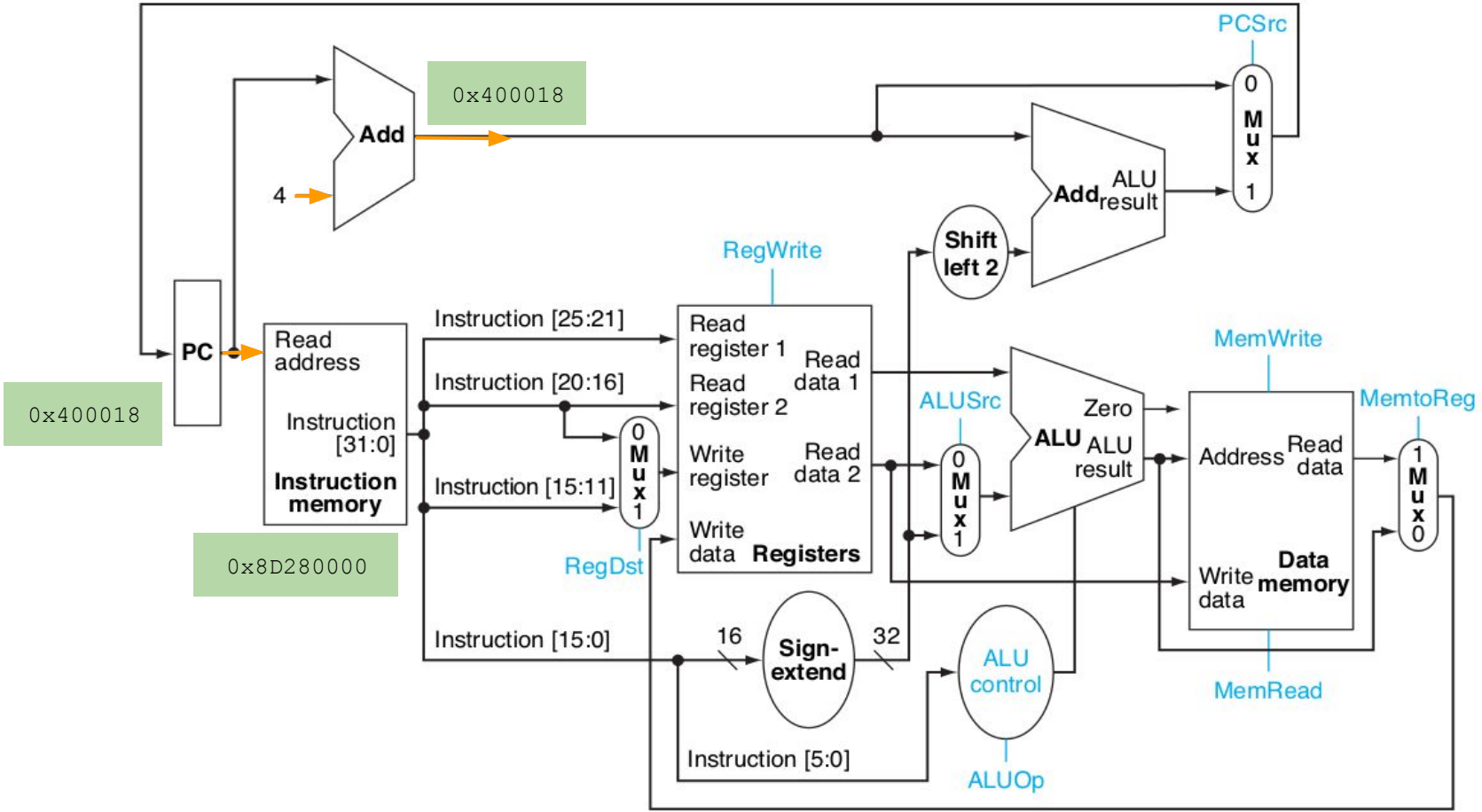
mnemônico: lw $t0, 0($t1)
endereço da instrução: 0x00400018
hex: 0x8D280000

Antes da busca, PC está com o endereço 0x400014

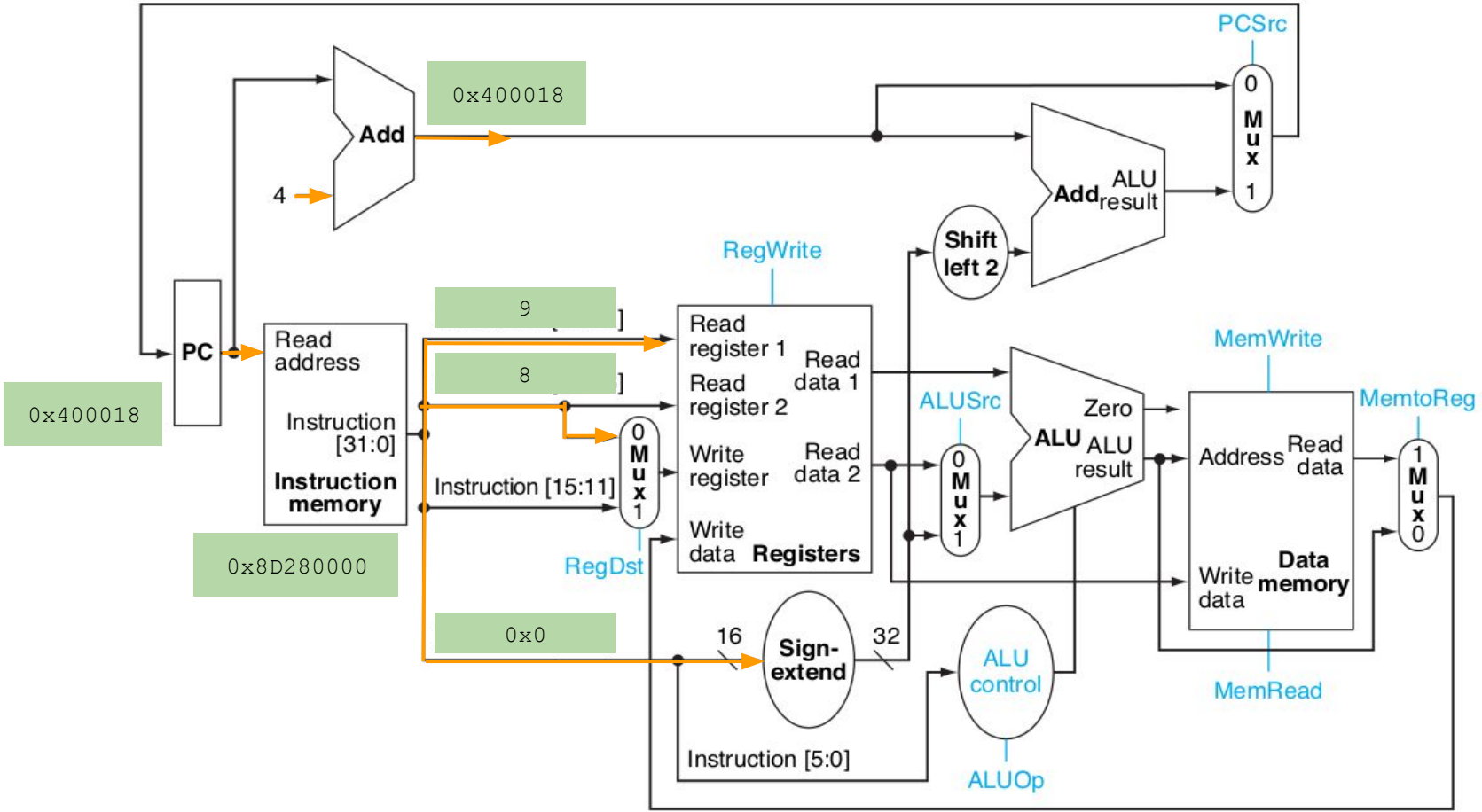
0x400014



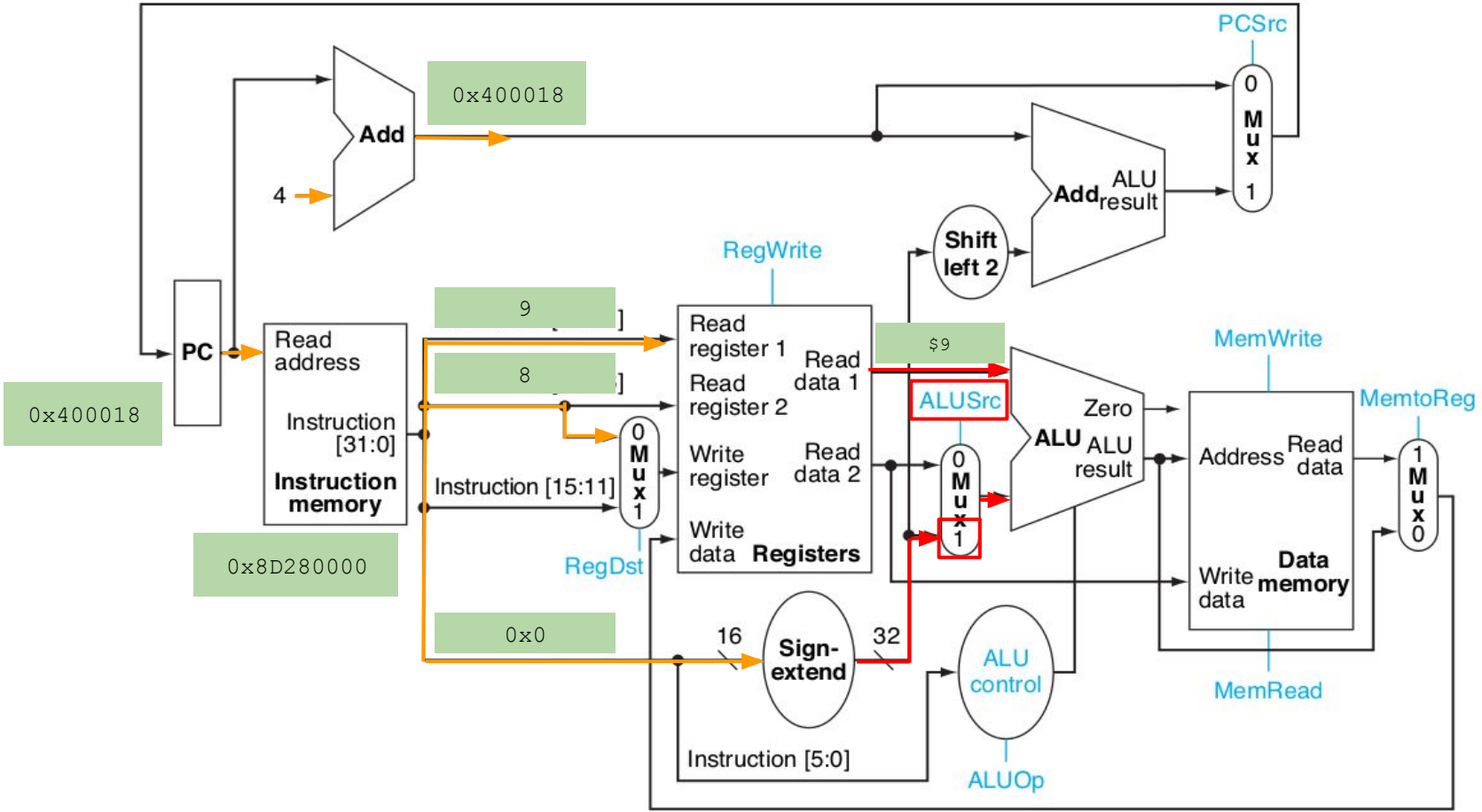
PC é incrementado em 4, e a instrução em 0x400018 é buscada



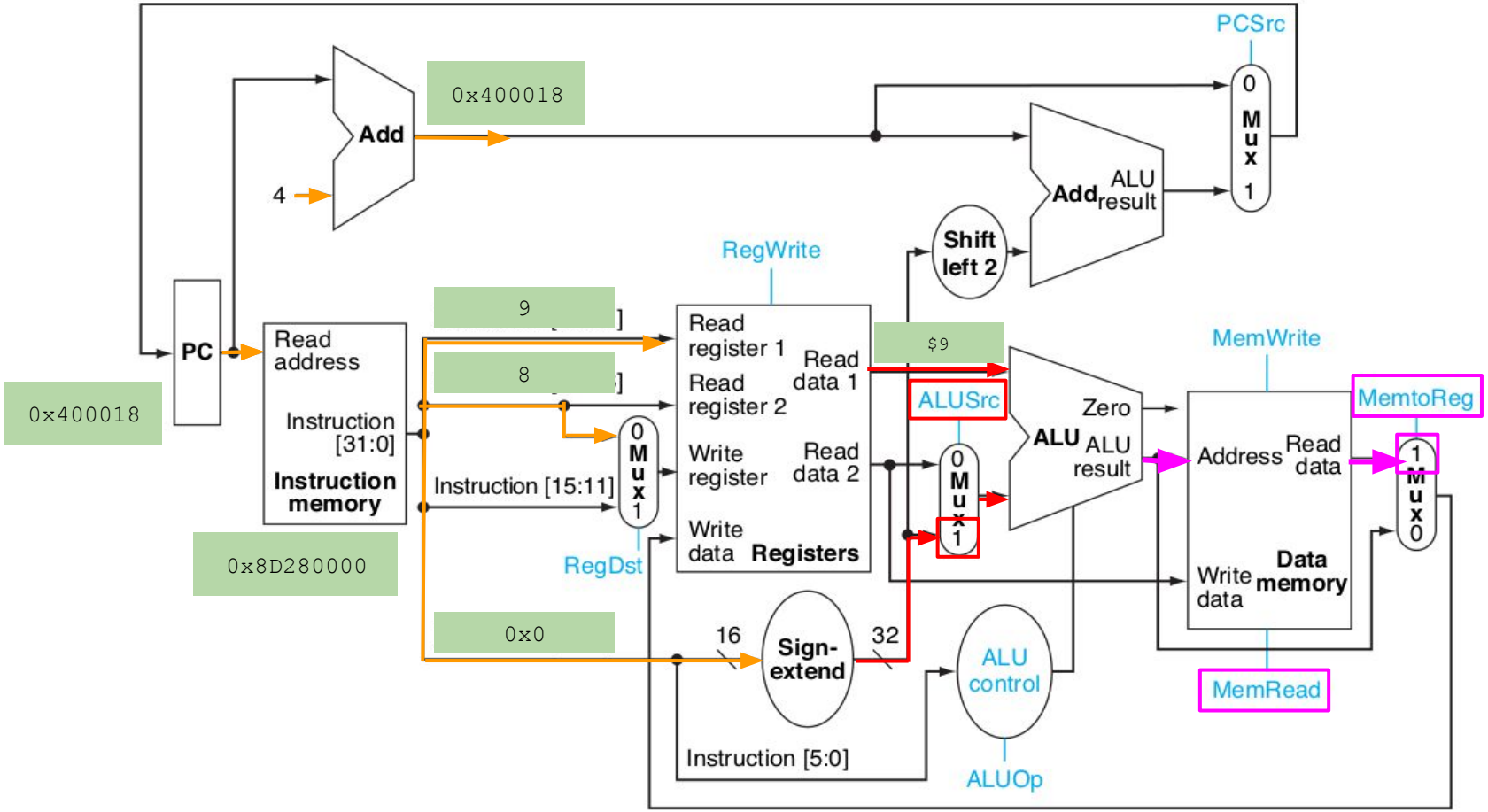
A instrução é decodificada, os sinais de controle são enviados



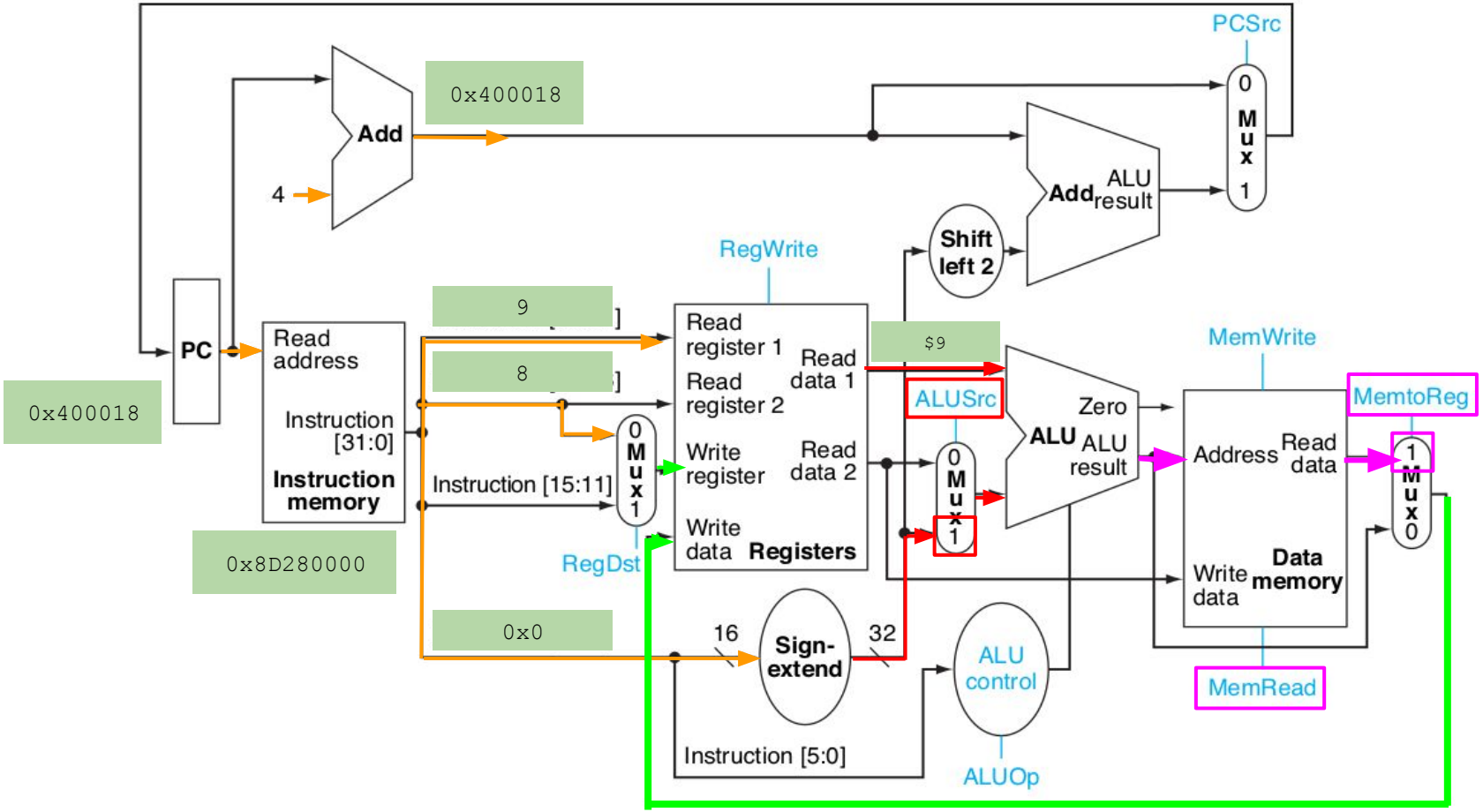
ALUSrc deve ser 1 para o cálculo do endereço



O endereço é acessado, com os sinais MemtoReg e MemRead



Os dados lidos são escritos no registrador destino



Instruções de desvio

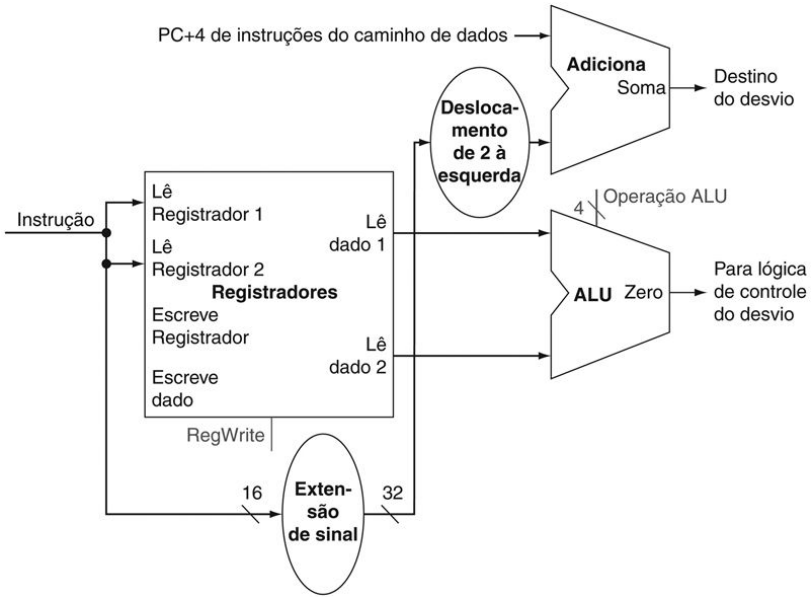
Lê operandos dos registradores

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 PC+4
  - Já calculado na busca da instrução



Exemplo de instrução de desvio

0x0040001c beq $t0, $s5, Exit (bne $8,$21,0x00000002)																if(R[rs]!=R[rt]) PC=PC+4+BranchAddr (0x05)														
0x04						8					21																	0x0002		
0	0	0	1	0	0	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				1				1				5				0				0				0				2		

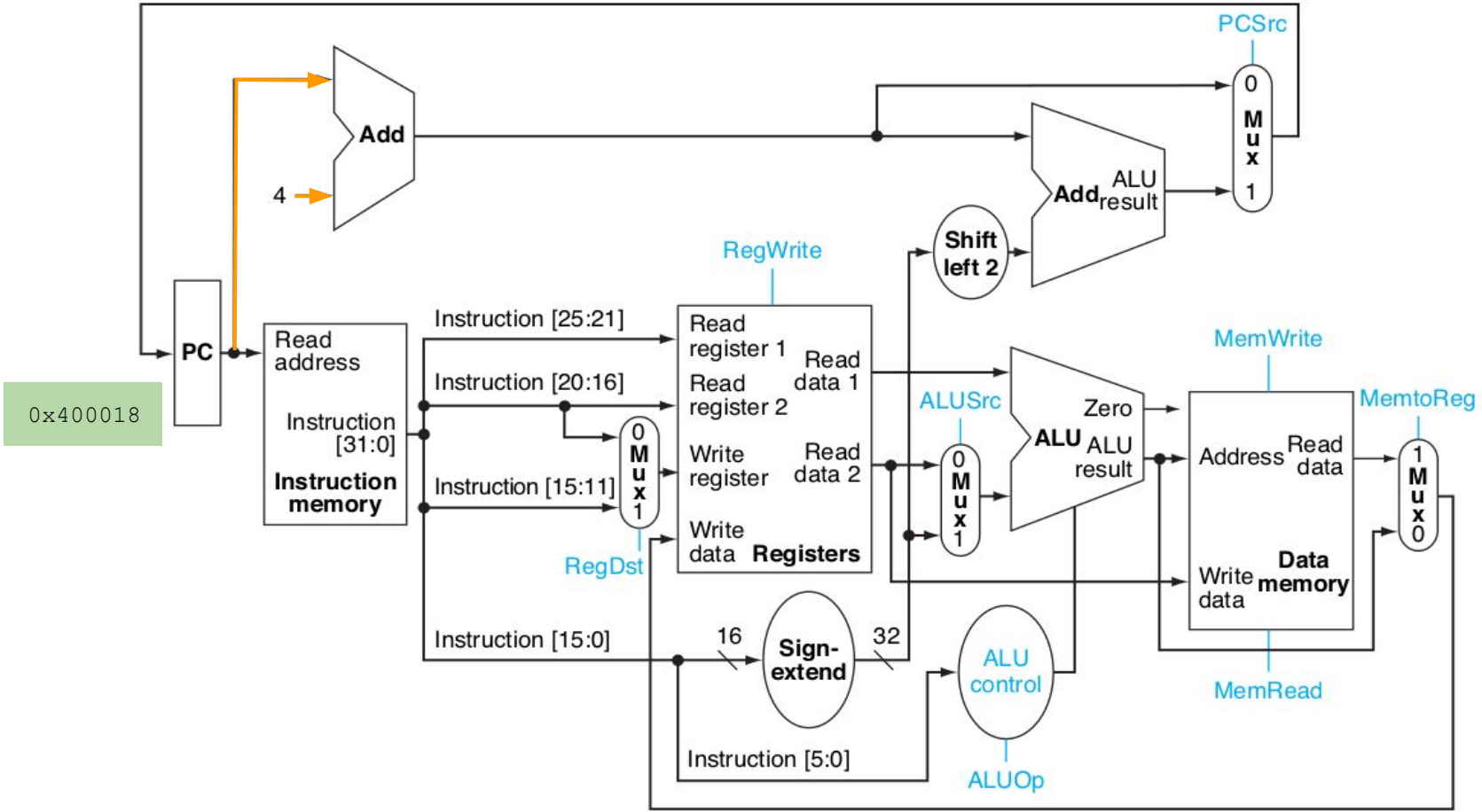
Tipo I

opcode	rs	rt	endereço
4	8	21	0x2

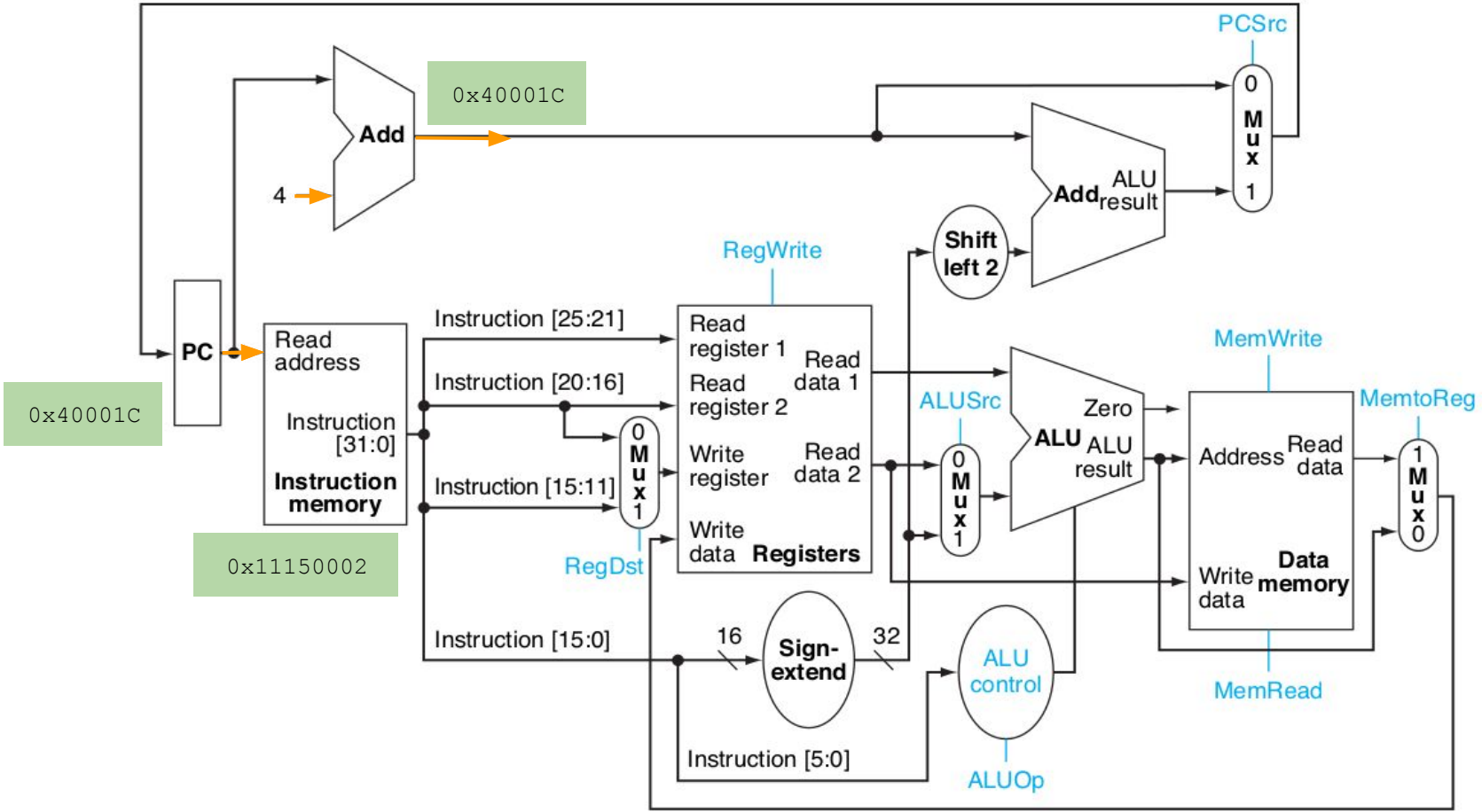
mnemônico: bne $t0, $s5, Exit
endereço da instrução: 0x0040001C
hex: 0x15150002

Antes da busca, PC está com o endereço 0x400018

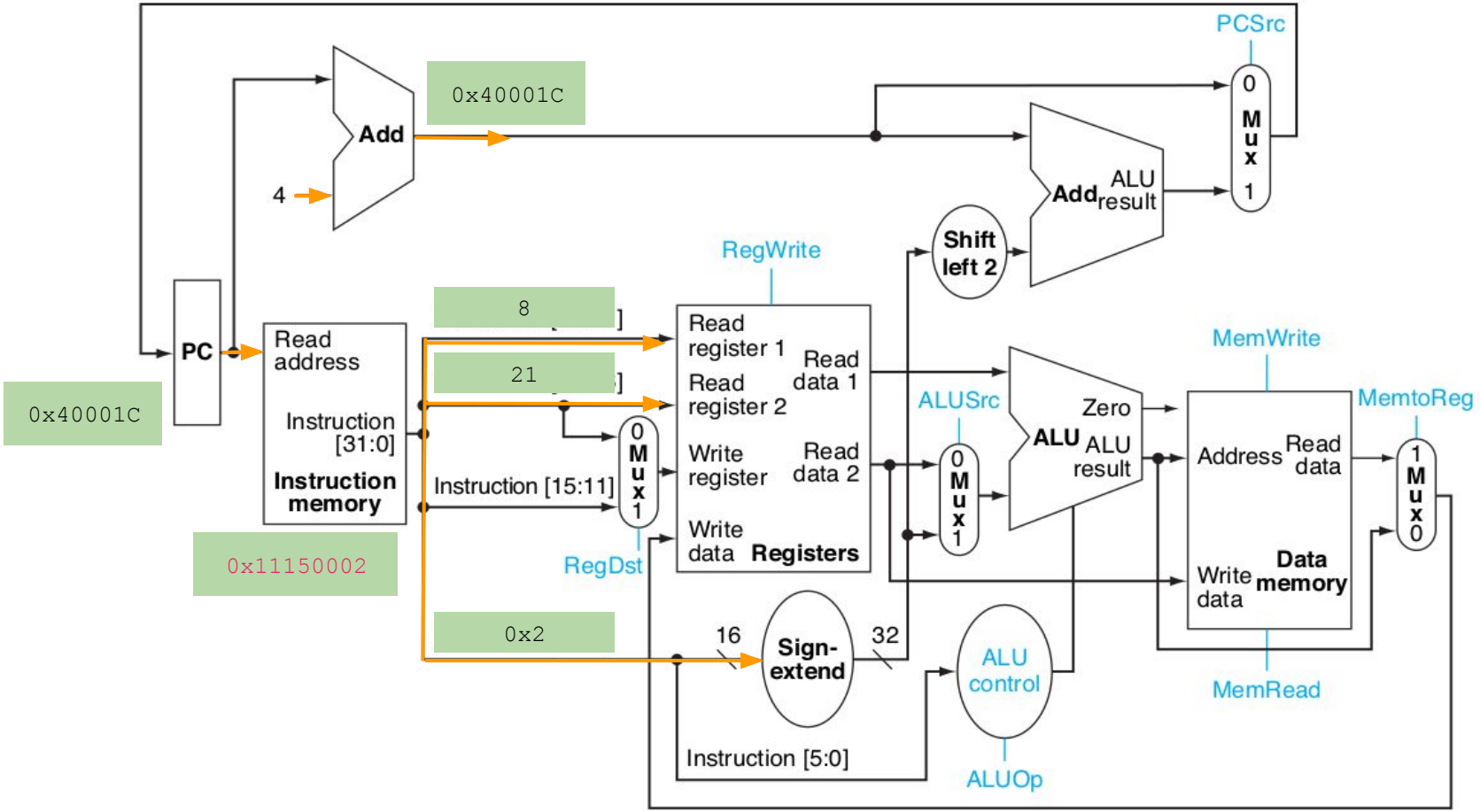
0x400018



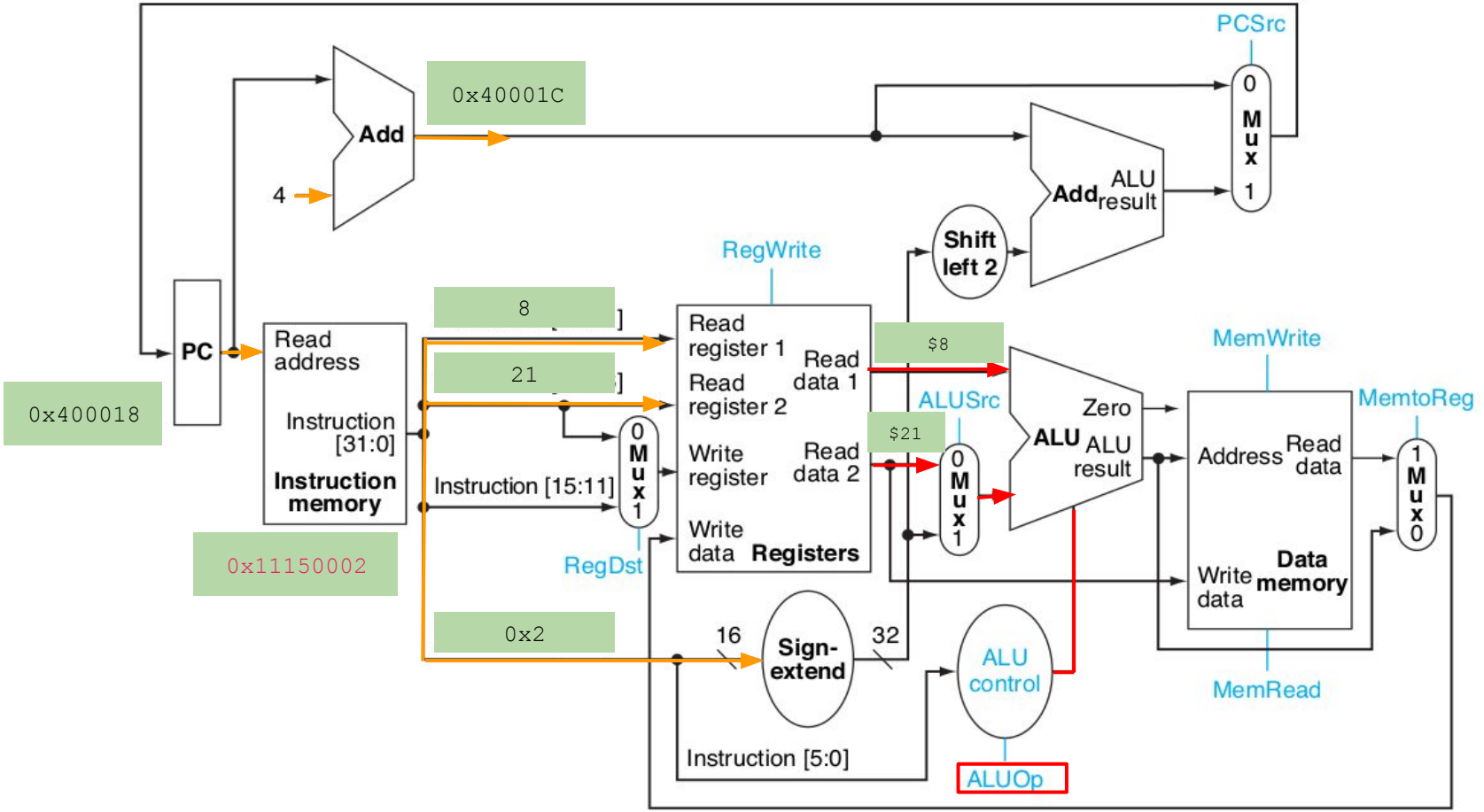
PC é incrementado em 4, e a instrução em 0x40001C é buscada



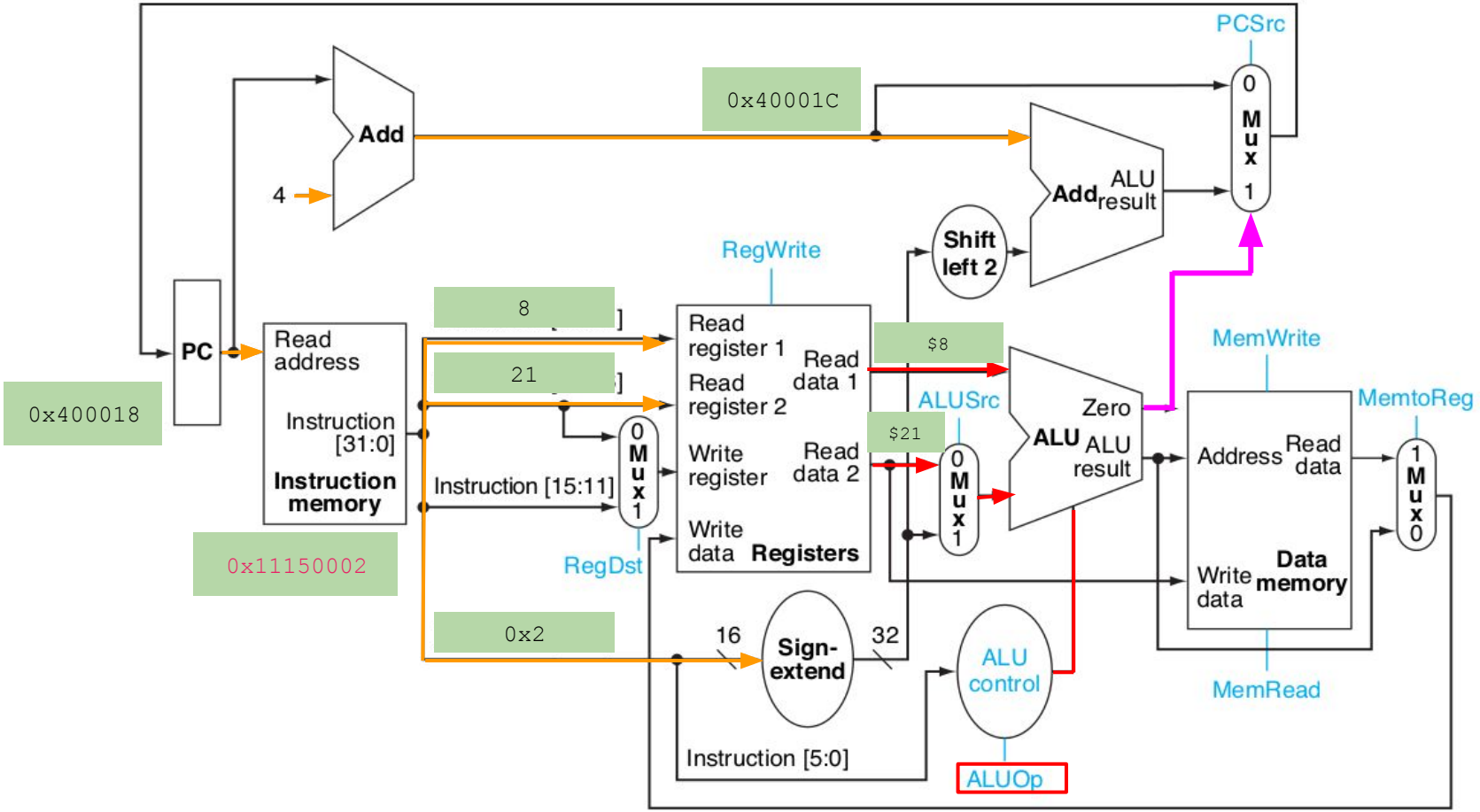
A instrução é decodificada, os sinais de controle são enviados



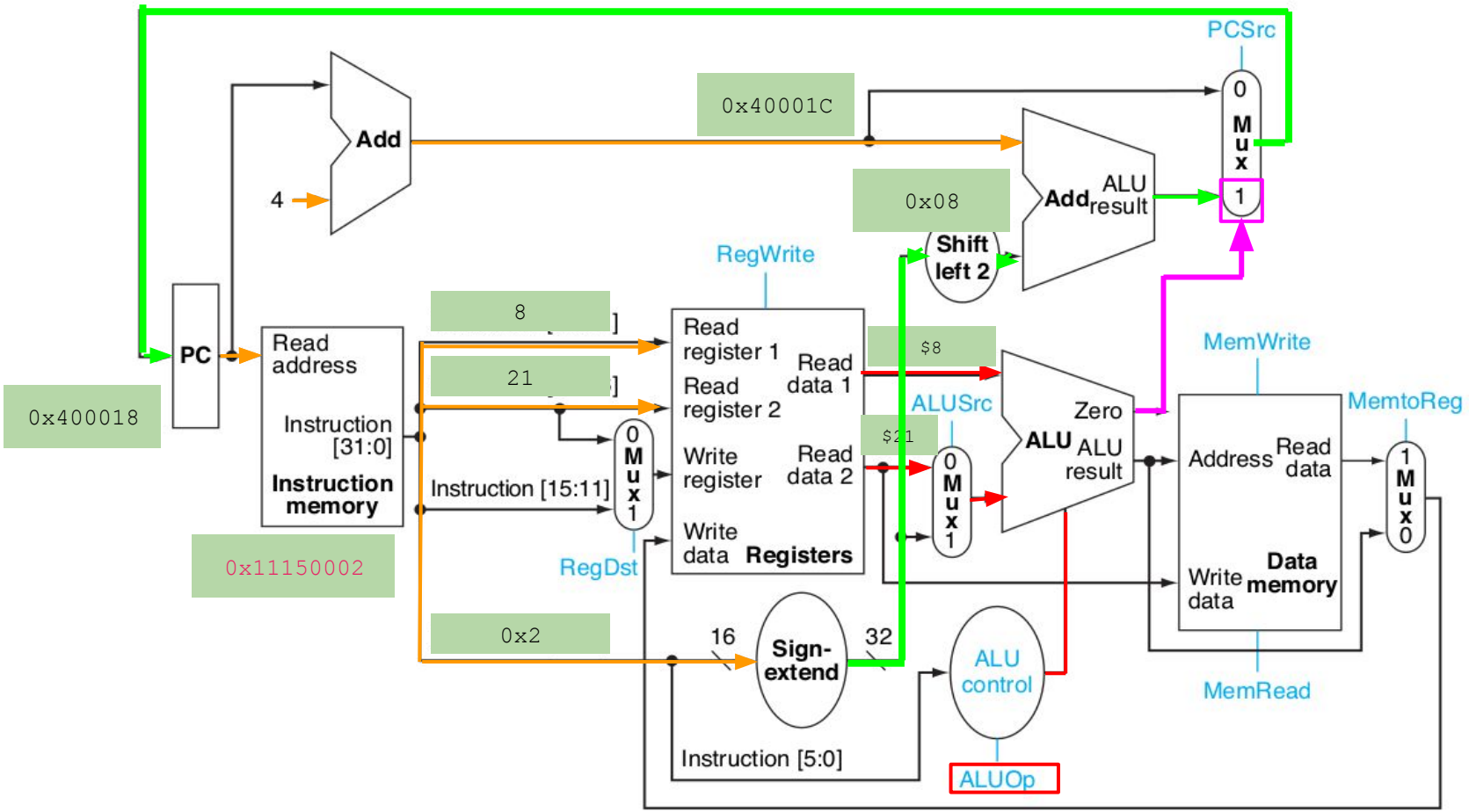
Os valores são lidos e a operação sub é enviada em ALUOp



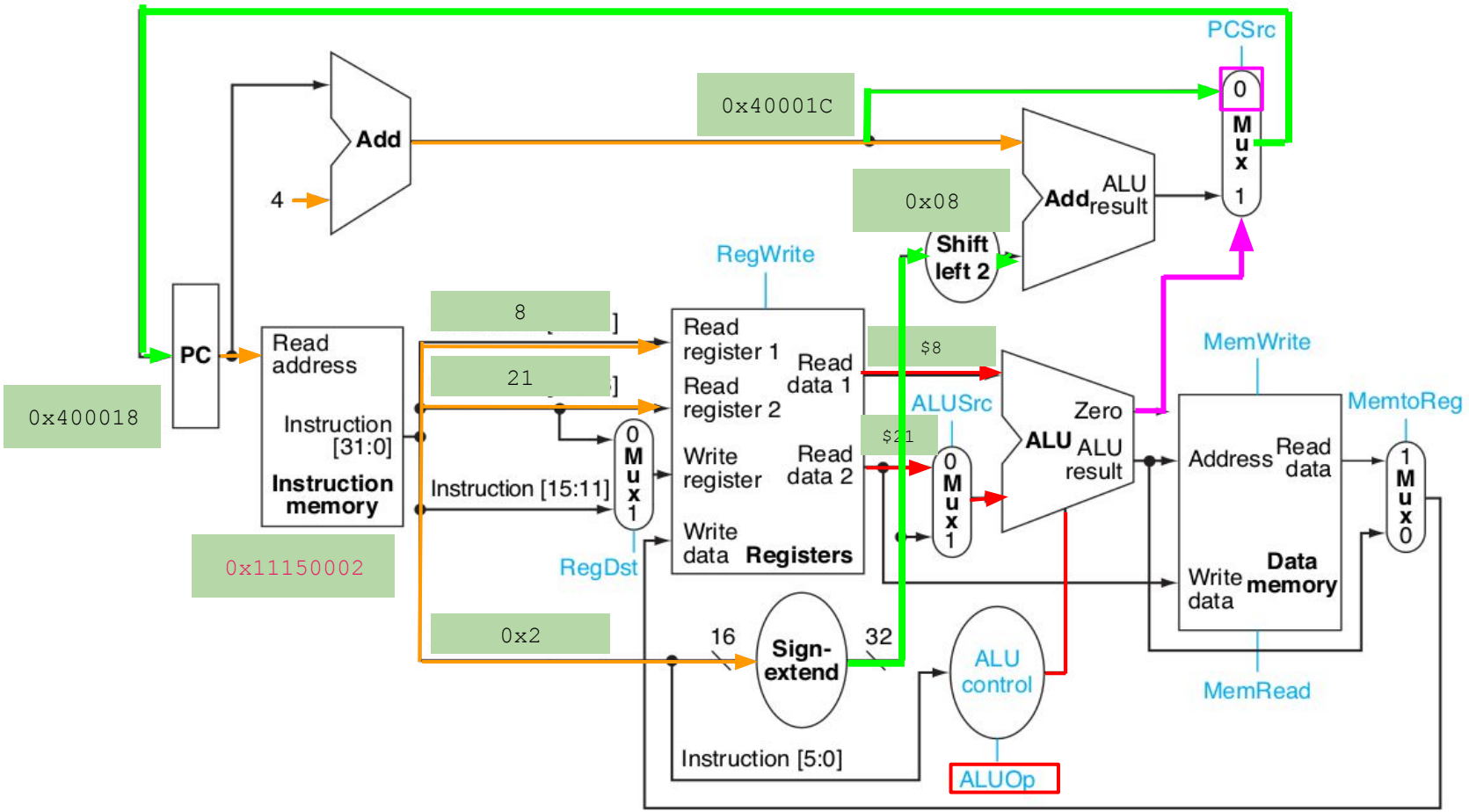
O detector de zero é enviado como PCSrc



Se o resultado for zero (beq), o novo valor de PC é PC+0x08



Se o resultado não for zero (beq), PC segue com PC+4



Compondo os elementos

O caminho de dados inicializado faz uma instrução em um ciclo de clock

- Cada elemento do datapath pode executar apenas uma função por vez
- Portanto, precisamos de instruções separadas para memórias e dados

Usar multiplexadores onde fonte de dados alternativas são usadas para instruções diferentes

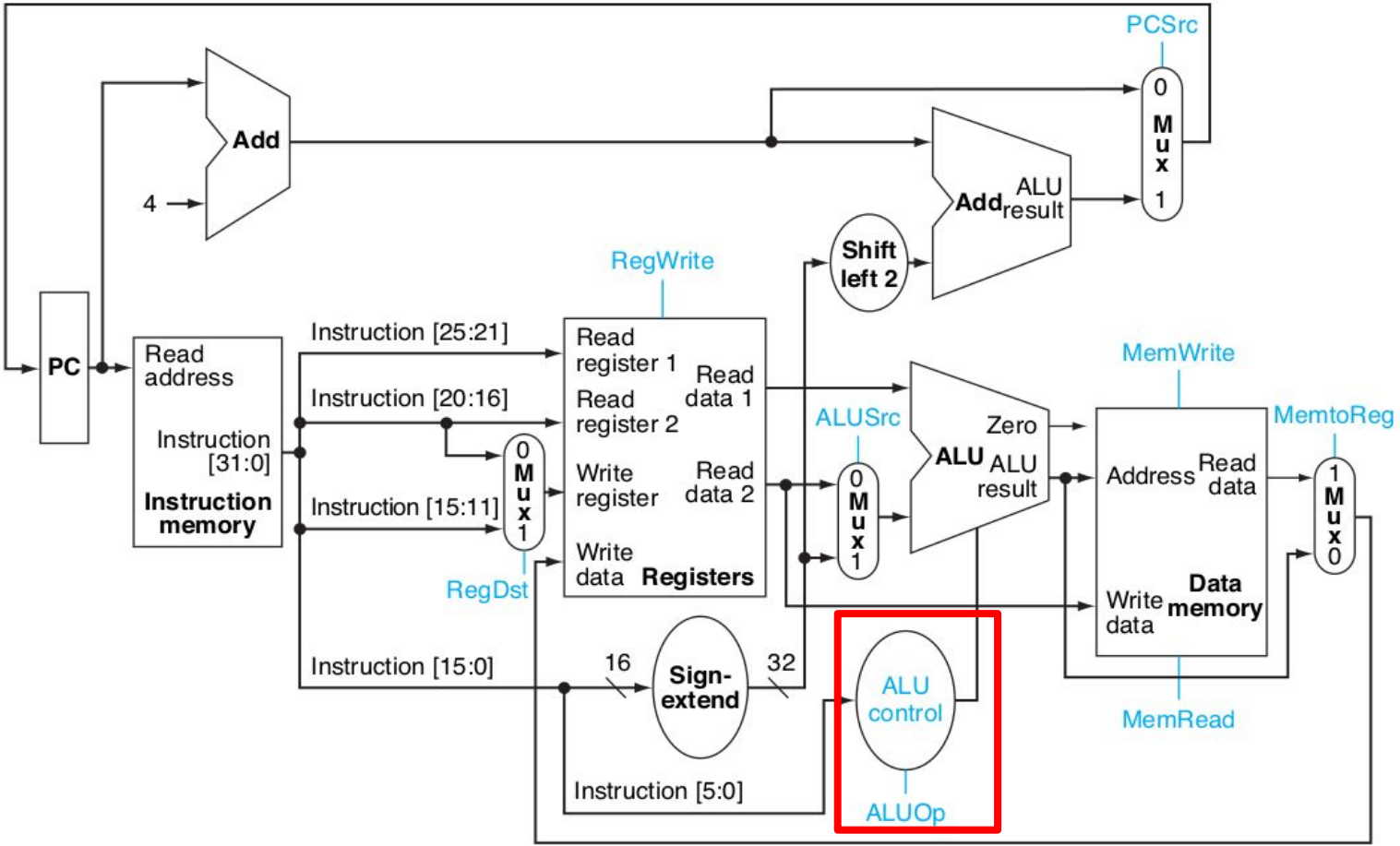
Controle da ULA

ULA é utilizada nas instruções

- load/store: função soma
- Desvio: função subtração
- Tipo R: depende do campo funct na instrução

Linhas controle ULA	Função
0000	AND
0001	OR
0010	add
0110	subtract
0111	set-on-less-than
1100	NOR

Sinal ALUOp


PCSrc
Add
4
M
u
x
0
1
ALU
Add result
Shift
left 2
RegWrite
PC
Read
address
Instruction
[31:0]
Instruction
memory
Instruction [25:21]
Instruction [20:16]
Instruction [15:11]
Instruction [15:0]
Instruction [5:0]
Read
register 1
Read
register 2
Write
register
Write
data
Read
data 1
Read
data 2
Registers
RegDst
ALUSrc
Zero
ALU
ALU
result
MemWrite
MemtoReg
Address
Read
data
Write
data
Data
memory
MemRead
16
Sign-
extend
32
ALU
control
ALUOp

ALU Control

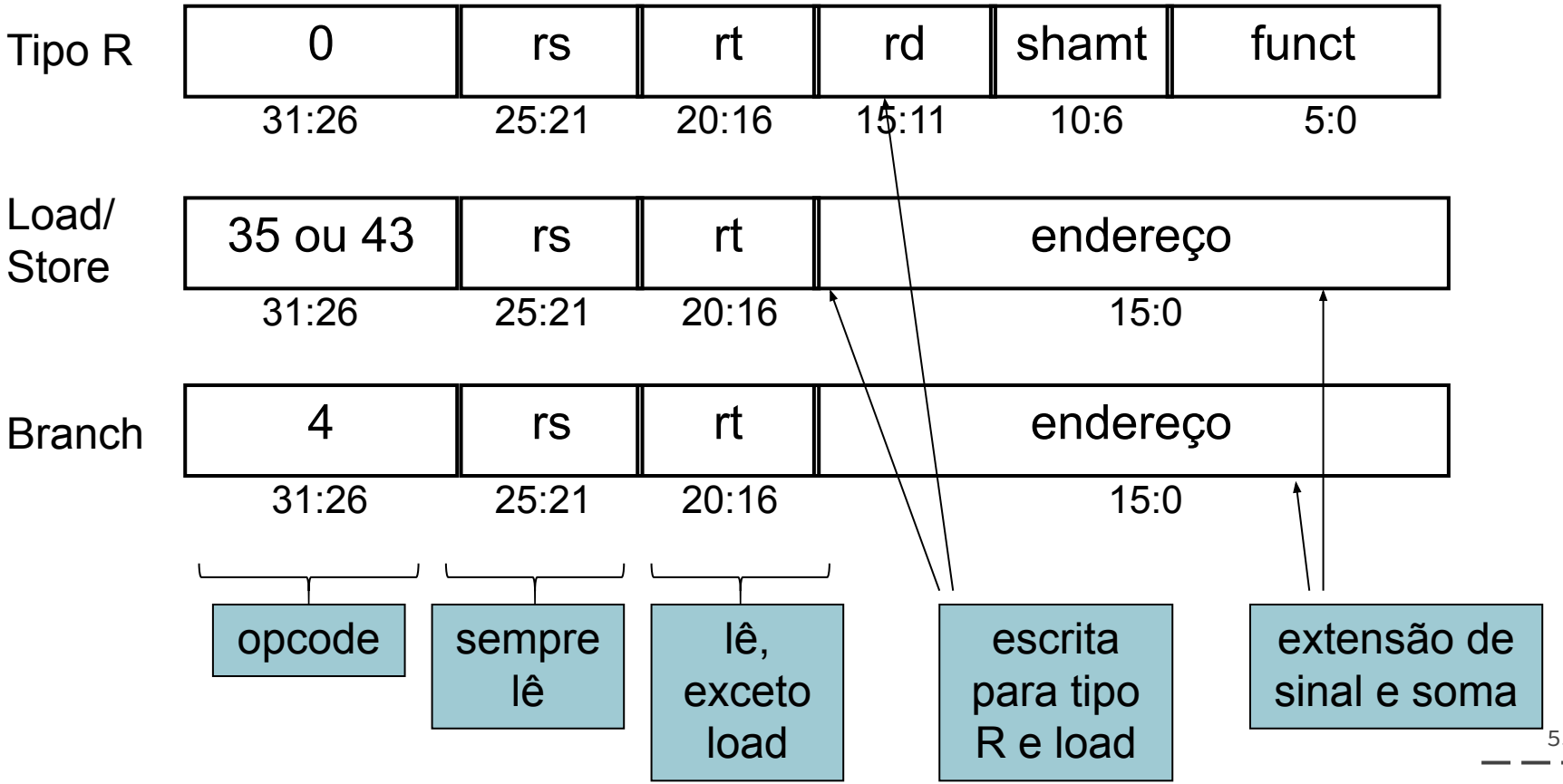
Suponha 2 bits ALUOp derivados do opcode

- A lógica combinacional deriva o controle da ULA

Opcode da instrução	OpALU	Operação da instrução	Campo funct	Ação da ALU desejada	Entrada do controle da ALU
LW	00	load word		add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
tipo R	10	add	100000	add	0010
tipo R	10	subtract	100010	subtract	0110
tipo R	10	AND	100100	AND	0000
tipo R	10	OR	100101	OR	0001
tipo R	10	set on less than	101010	set on less than	0111

Unidade de controle principal

Deriva os sinais de controle com base na instrução



Sinais de controle

Nome do sinal	Efeito quando inativo	Efeito quando ativo
RegDst	O número do registrador destino para entrada Registrador para escrita vem do campo rt (bits 20:16).	O número do registrador destino para a entrada Registrador para escrita vem do campo rd (bits 15:11).
EscreveReg	Nenhum.	O registrador na entrada Registrador para escrita é escrito com o valor da entrada Dados para escrita.
OrigALU	O segundo operando da ALU vem da segunda saída do banco de registradores (Dados da leitura 2).	O segundo operando da ALU consiste nos 16 bits mais baixos da instrução com sinal estendido.
OrigPC	O PC é substituído pela saída do somador que calcula o valor de PC + 4.	O PC é substituído pela saída do somador que calcula o destino do desvio.
LeMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é colocado na saída Dados da leitura.
EscreveMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é substituído pelo valor na entrada Dados para escrita.
MemparaReg	O valor enviado à entrada Dados para escrita do banco de registradores vem da ALU.	O valor enviado à entrada Dados para escrita do banco de registradores vem da memória de dados.

Sinais de controle

RegDst

WriteReg

ALUSrc

PCSrc

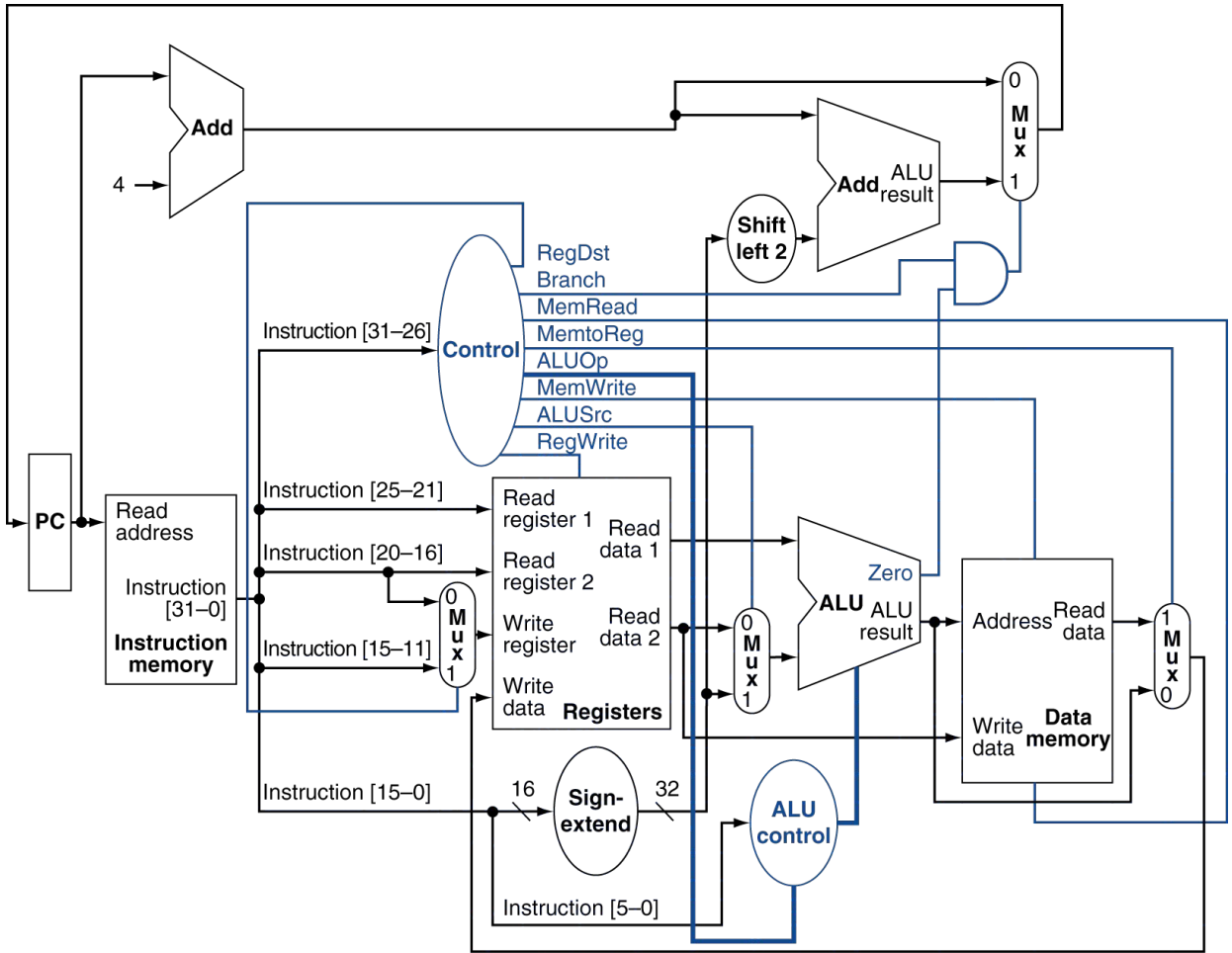
MemRead

MemWrite

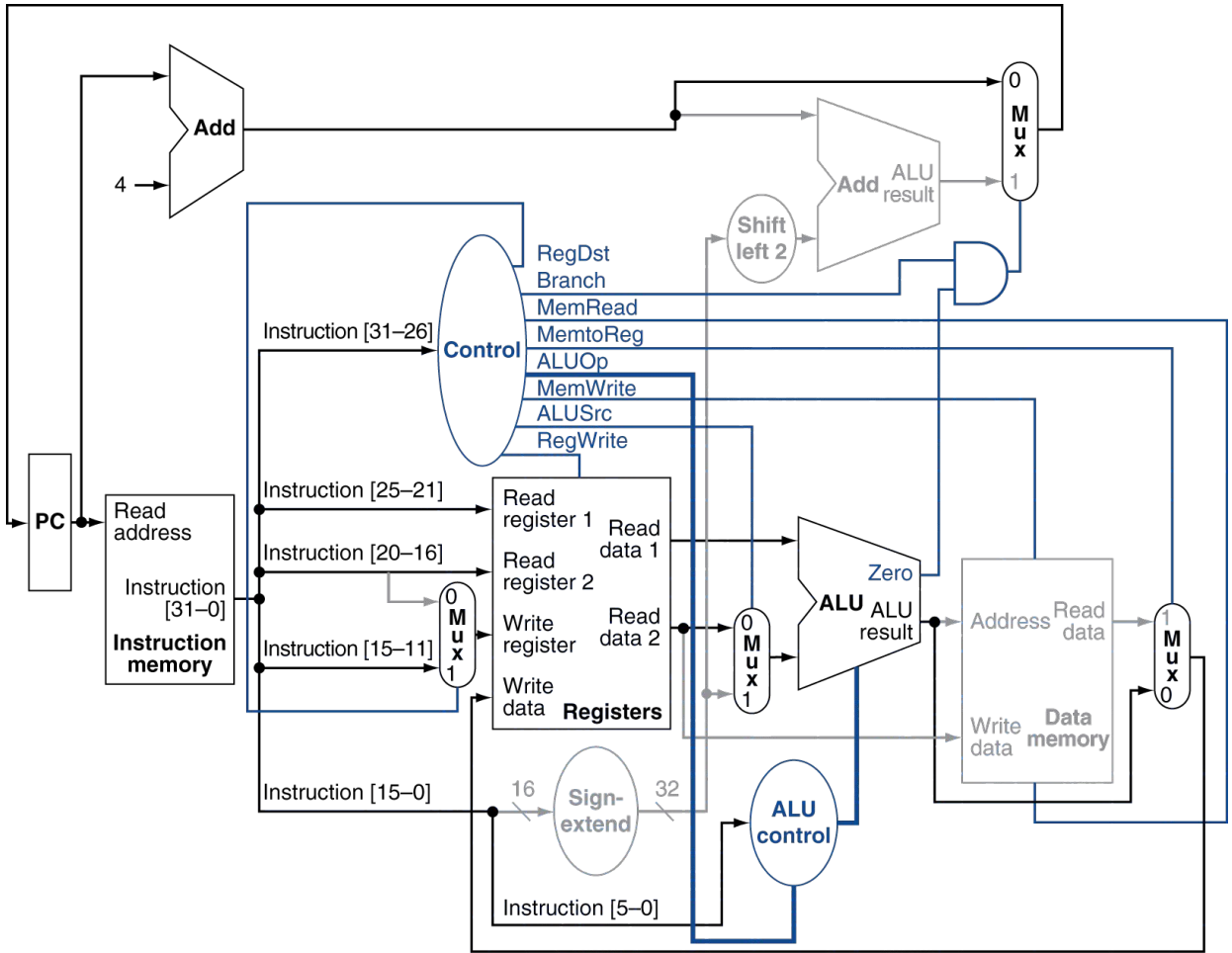
MemtoReg

Nome do sinal	Efeito quando inativo	Efeito quando ativo
RegDst	O número do registrador destino para entrada Registrador para escrita vem do campo rt (bits 20:16).	O número do registrador destino para a entrada Registrador para escrita vem do campo rd (bits 15:11).
EscreveReg	Nenhum.	O registrador na entrada Registrador para escrita é escrito com o valor da entrada Dados para escrita.
OrigALU	O segundo operando da ALU vem da segunda saída do banco de registradores (Dados da leitura 2).	O segundo operando da ALU consiste nos 16 bits mais baixos da instrução com sinal estendido.
OrigPC	O PC é substituído pela saída do somador que calcula o valor de PC + 4.	O PC é substituído pela saída do somador que calcula o destino do desvio.
LeMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é colocado na saída Dados da leitura.
EscreveMem	Nenhum.	O conteúdo da memória de dados designado pela entrada Endereço é substituído pelo valor na entrada Dados para escrita.
MemparaReg	O valor enviado à entrada Dados para escrita do banco de registradores vem da ALU.	O valor enviado à entrada Dados para escrita do banco de registradores vem da memória de dados.

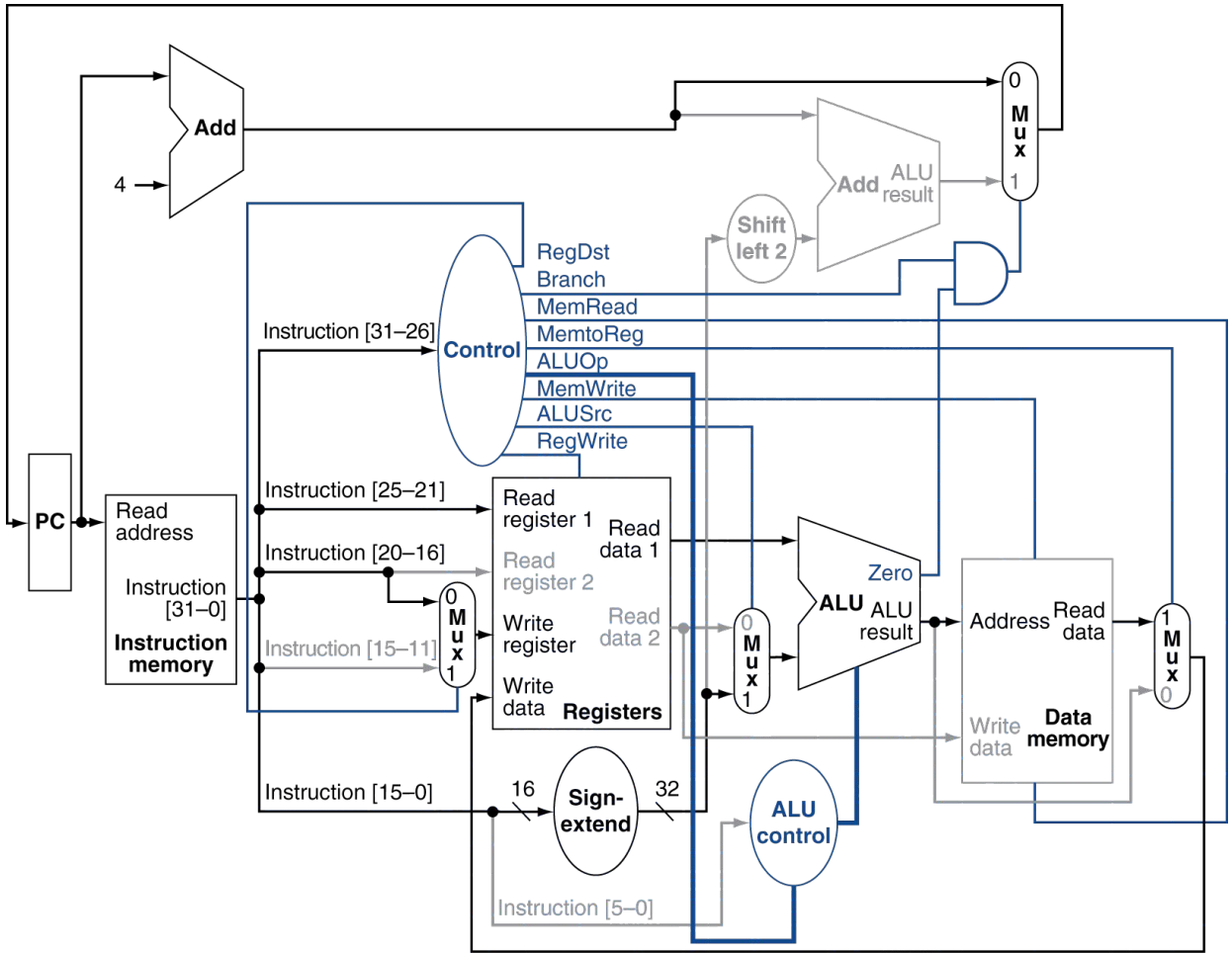
Datapath com controle



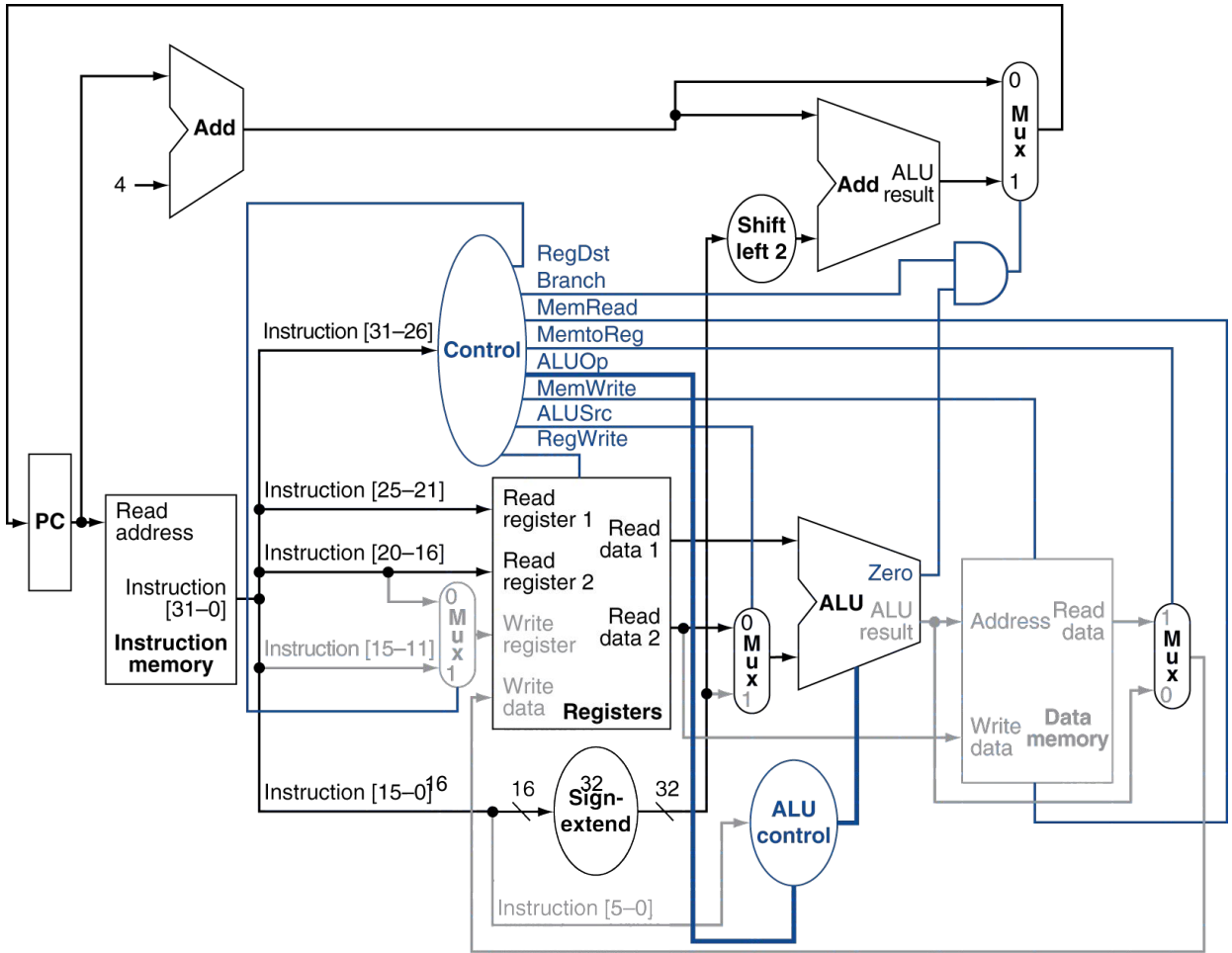
Instruções R



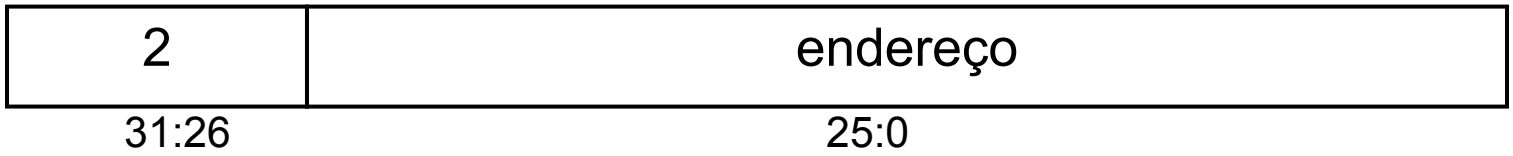
Instrução Load



Instrução BEQ



Implementando jumps



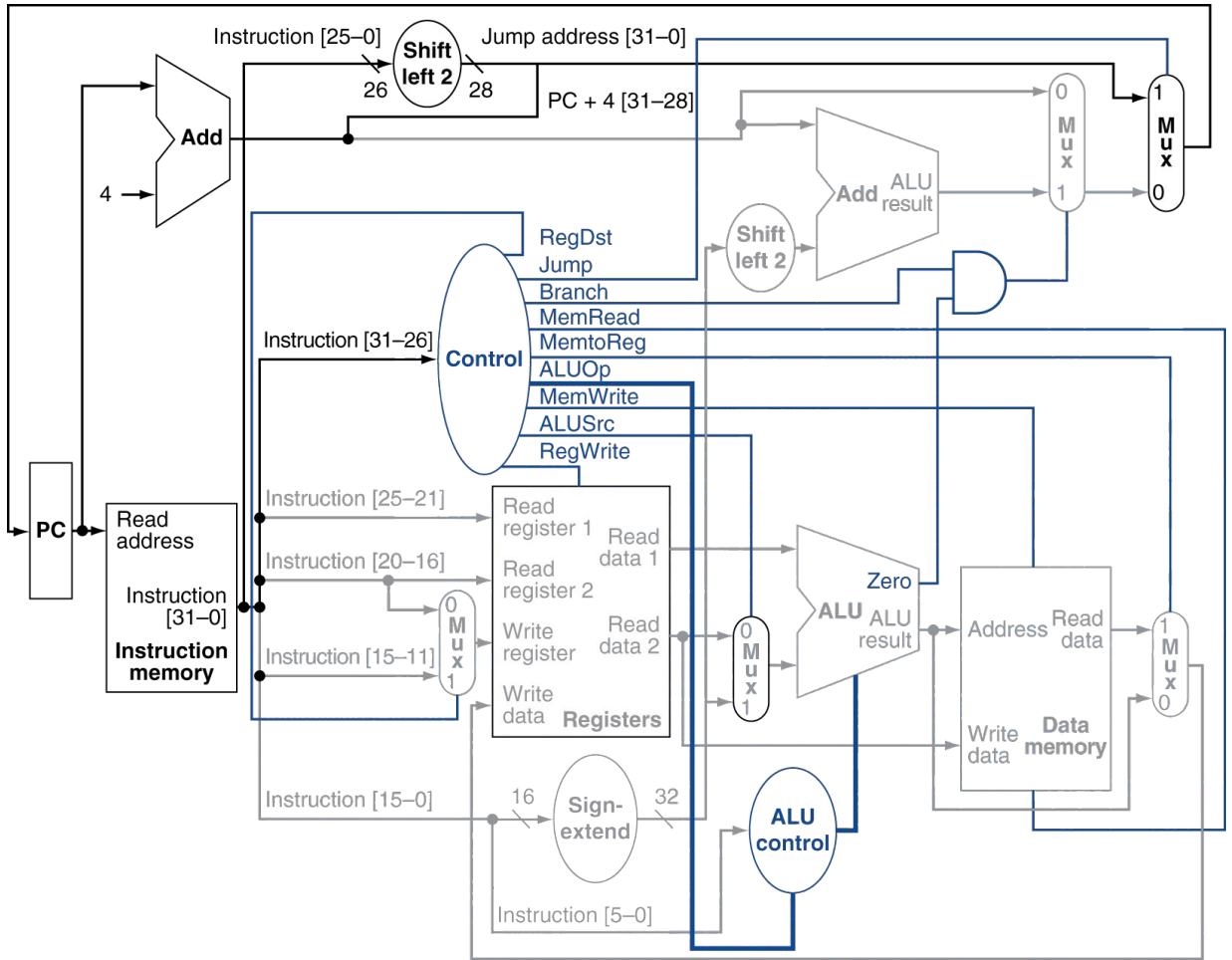
Jumps usam endereços de palavras

Atualizam PC concatenando

- 4 bits mais significativos do PC antigo
- 26 bits do endereço de jump
- 00

Precisam de um sinal extra de controle decodificado do opcode

Jump


Instruction [25–0]
Shift left 2
Jump address [31–0]
26
28
PC + 4 [31–28]
Add
4
0
M
u
x
1
1
M
u
x
0
Add
ALU result
Shift left 2
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign-extend
32
Zero
ALU
ALU result
ALU control
Address
Read data
Write data
Data memory

Capítulo 4 - O processador

Exercícios sugeridos:

4.1, 4.2, 4.4

Referências

Materiais disponibilizados pelos professores Lucas Wanner, Paulo Gonçalves, Ricardo Pannain e Rogério Ap. Gonçalves.

Patterson, David A. Hennessy, John L. Organização e Projeto de Computadores. Disponível em: Minha Biblioteca, (5a. edição). Grupo GEN, 2017.

Slides do livro PATTERSON, David A. e HENNESSY, John L. *Computer Organization and Design Risc-V Edition: The Hardware Software Interface*. Estados Unidos, Ed. Morgan Kauffman, 2020.