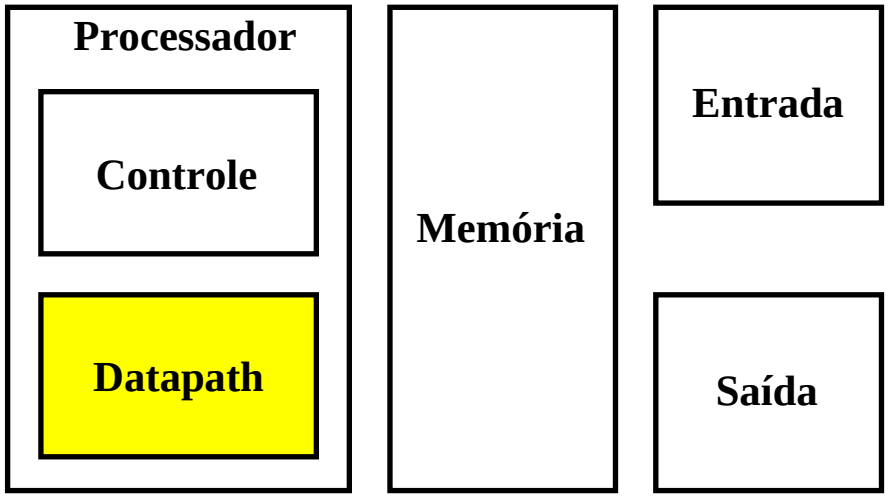
CPU: estrutura de um datapath



Motivação

$$\text{tempo}_{\text{CPU}} = I \times CPI \times T$$

	I	CPI	T
Compilador	**X**	**(X)**	
Organização		**X**	**X**

- **Implementação determina CPI e T**
- **Implementação de um subconjunto das instruções do MIPS**
- **lw, sw**
- **add, sub, and, or, slt**
- **beq, j**

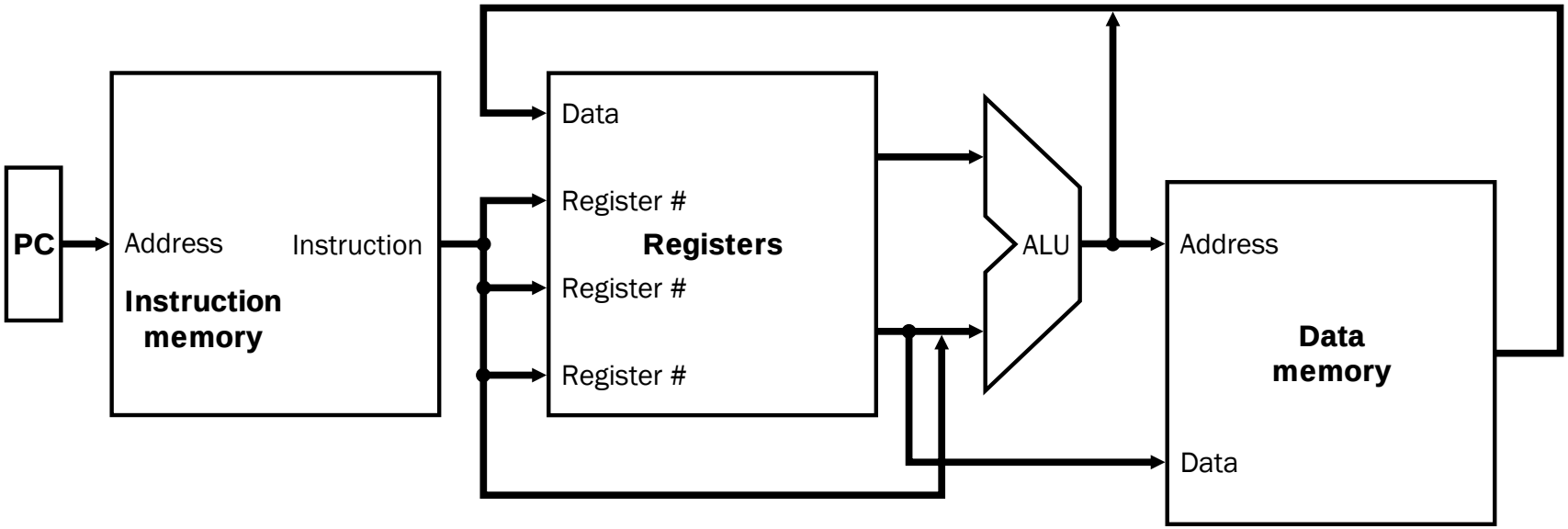
Etapas da execução de instruções

- **Enviar o valor no PC para a memória e buscar a instrução**
  - **Busca da instrução**
    - » **“Instruction fetch”**
- **Decodificar a instrução e ler operando(s)-fonte nos registradores indicados no formato**
  - **Decodificação da instrução**
    - » **“Instruction decoding”**
  - **Busca de operandos**
    - » **“Operand fetch”**

Etapas da execução de instruções

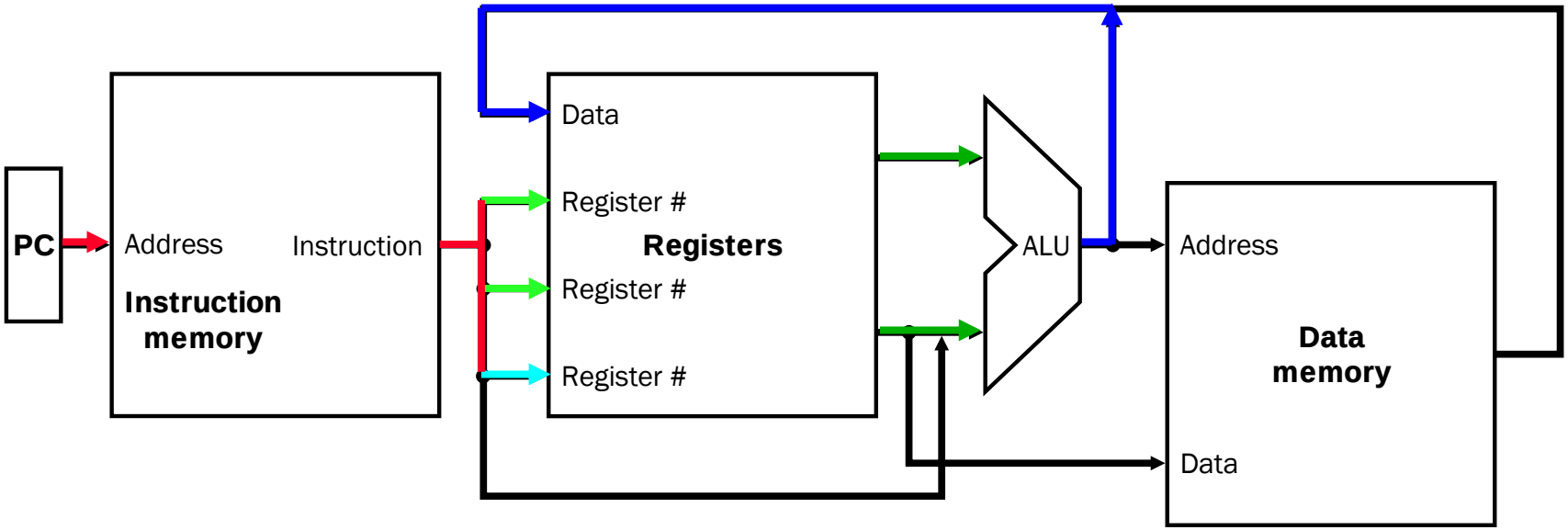
- **Efetuar cálculo na ALU**
  - **cálculo de endereço efetivo (lw, sw);**
  - **operação aritmética ou lógica (add, and, slt, etc.);**
  - **comparação (beq)**
- **Ler/escrever dados na memória ou escrever resultados em registrador**
  - **leitura da memória (lw),**
  - **escrita na memória (sw)**
  - **escrita em registradores (aritméticas e lógicas, lw)**
  - **alteração do PC (beq, j)**

Visão geral da implementação



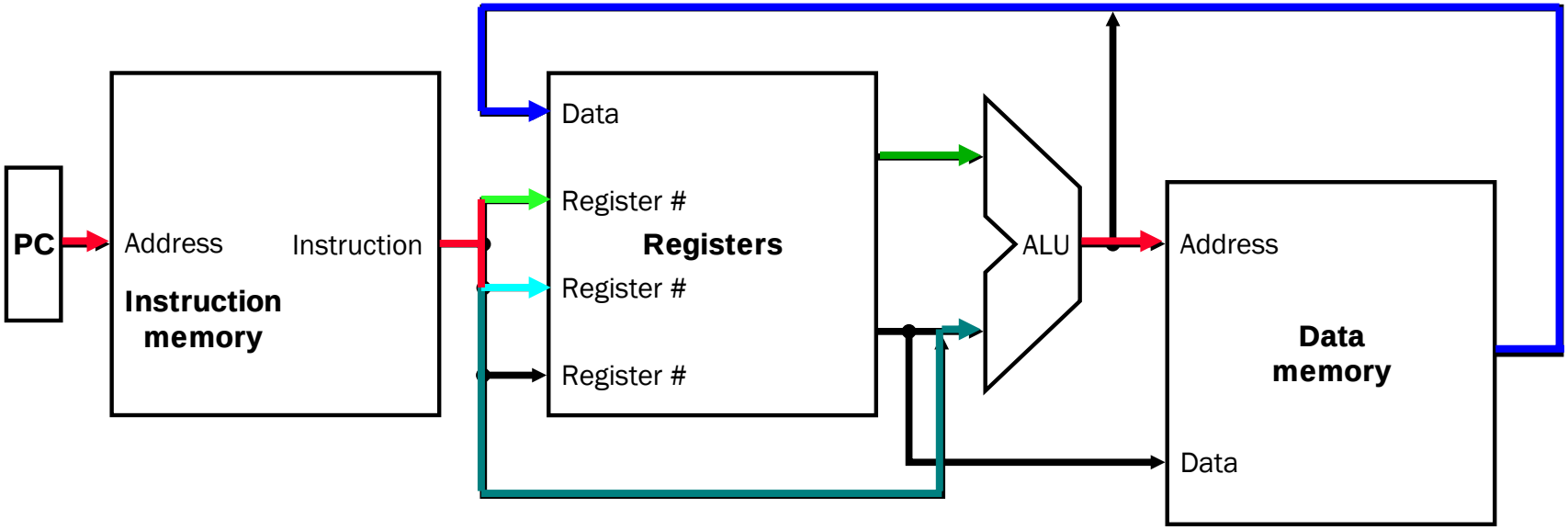
Visão geral da implementação

- **add $s1, $s2, $s3**



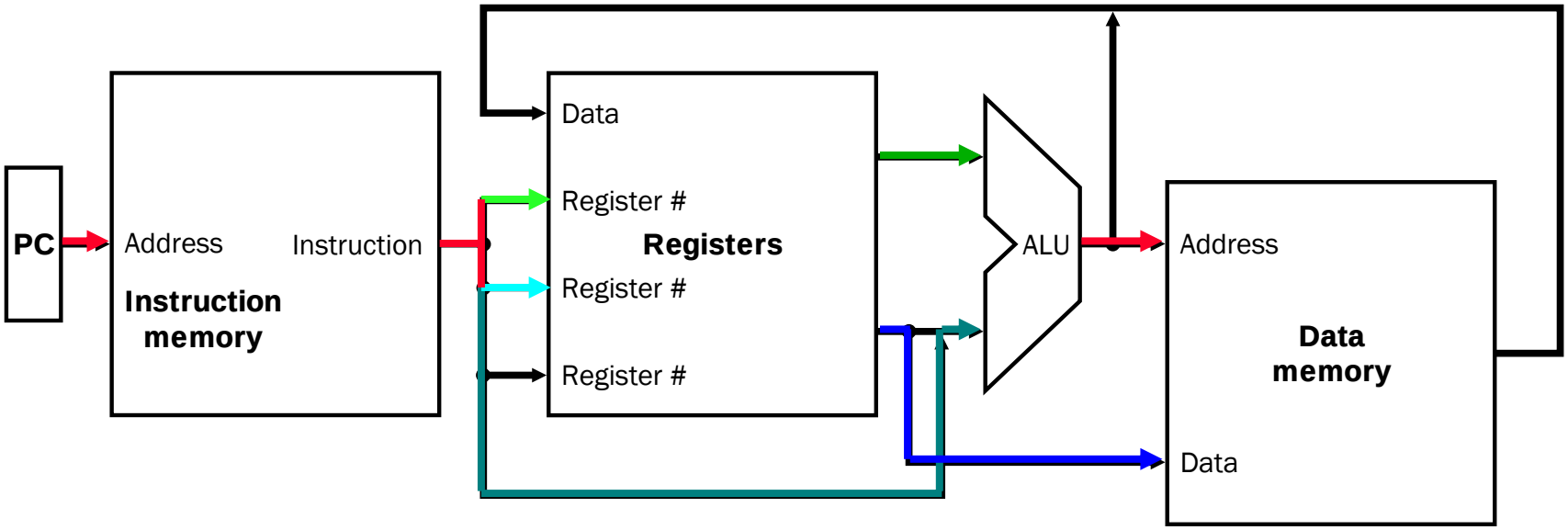
Visão geral da implementação

- **lw $s1, K($s2)**



Visão geral da implementação

- **sw $s1, K($s2)**

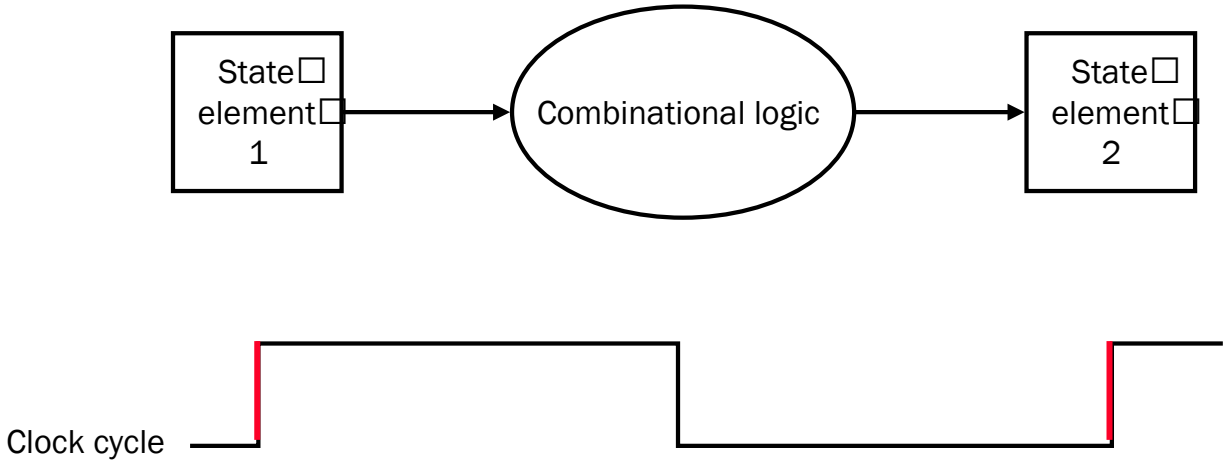


Componentes da implementação

- **Elementos combinacionais**
  - **saída depende do valor atual das entradas**
  - **Exemplo: ALU**
- **Elementos de armazenamento**
  - **Armazena um valor (estado)**
  - **Exemplo: memória, registradores**
  - **Entrada de dado, entrada de relógio, saída**
- **Circuito seqüencial**
  - **Saída depende da entrada e do estado**

Temporização

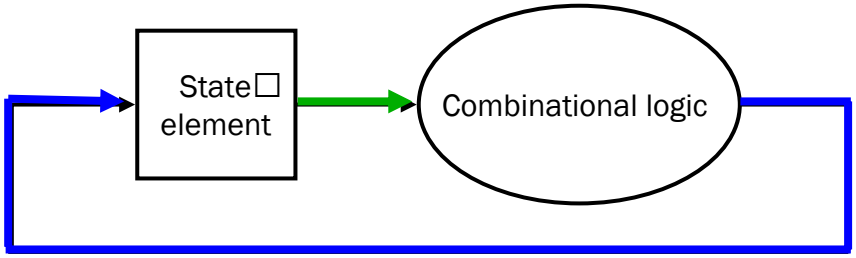
- **Quando sinais podem ser lidos/escritos ?**
  - **Sincronismo com um sinal de relógio**
  - **Hipótese: elementos de armazenamento são atualizados na transição do relógio**



Exemplo: addi $s1, $s2, #4

Temporização

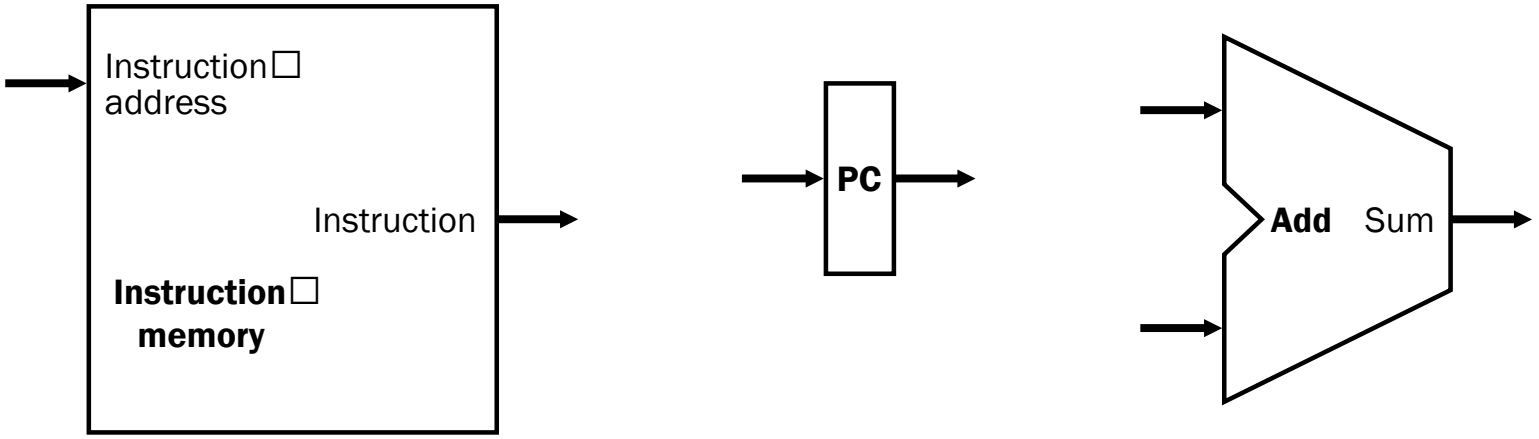
- **Laços de realimentação**
  - **Relógio quebra dependência mútua**
    - » **Saída depende da entrada (entre transições)**
    - » **Entrada depende da saída (amostragem na transição)**



Exemplo: addi $s1, $s1, #4

Suporte para busca de instruções

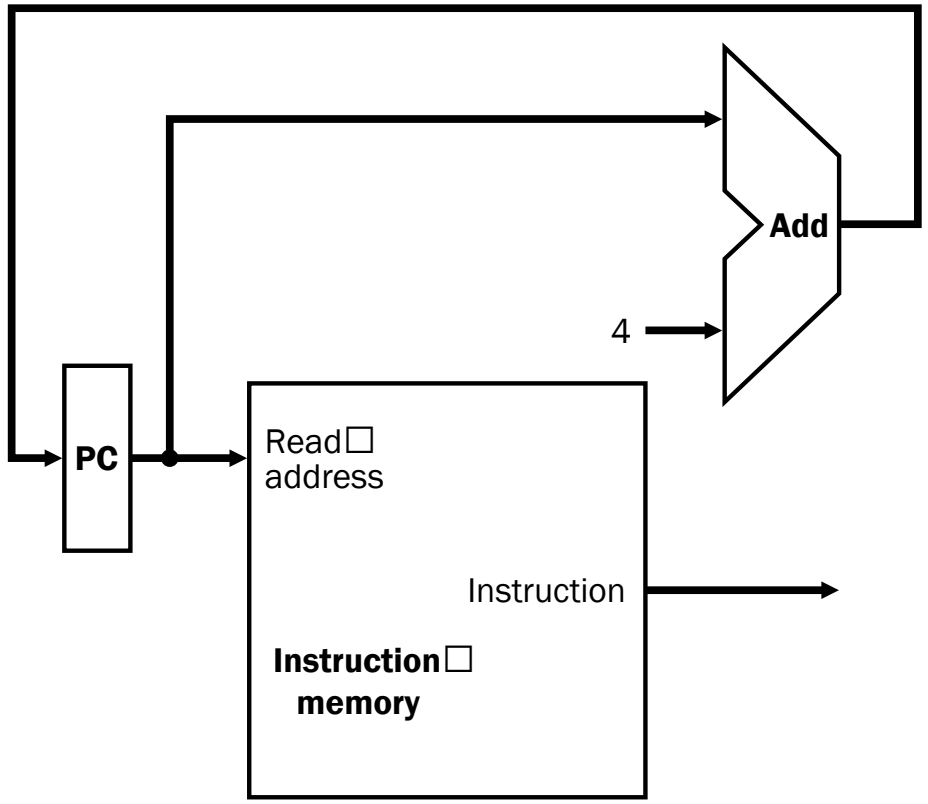
- **Armazenar instruções**
- **Buscar uma instrução**
- **Computar o endereço da próxima instrução**



a. Instruction memory b. Program counter c. Adder

Suporte para busca de instruções

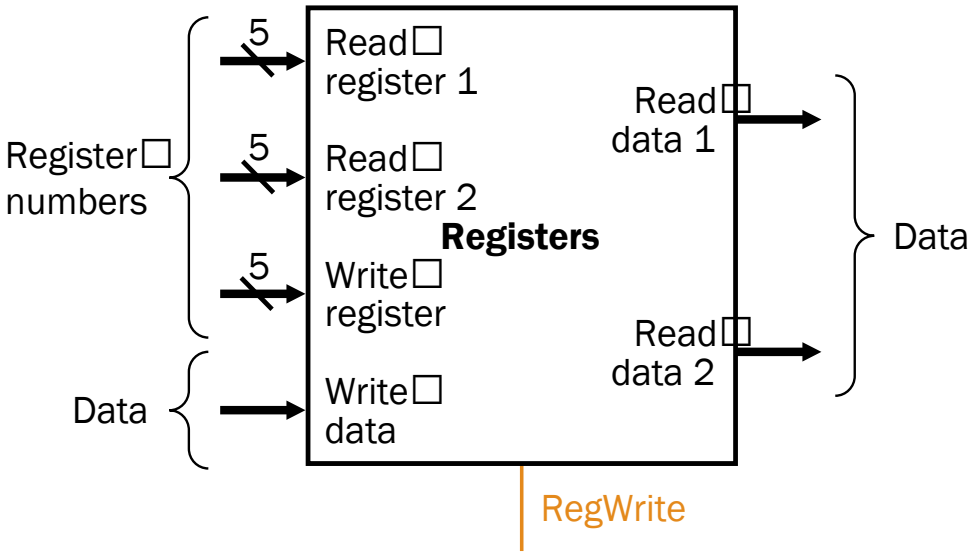
- **Endereçamento**
- **Acesso à instrução corrente**
- **Incremento do PC**



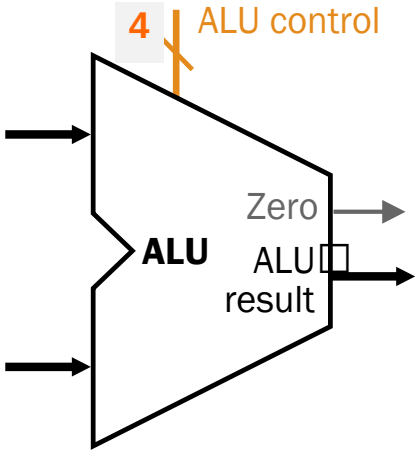
Suporte para instruções do tipo R

- **Banco de registradores**
  - **Portas: leitura (2) + escrita (1)**
- **Unidade lógica e aritmética**

add $t1,$t2,$t3



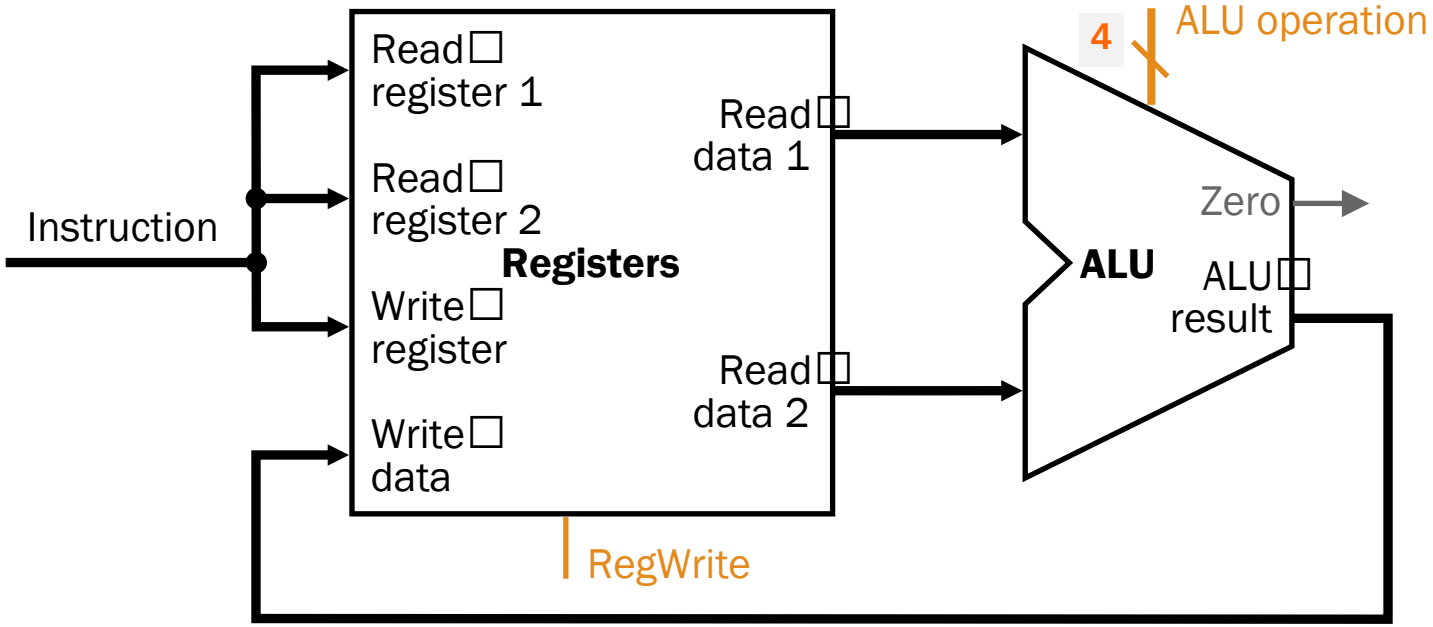
a. Registers



b. ALU

Suporte para instruções do tipo R

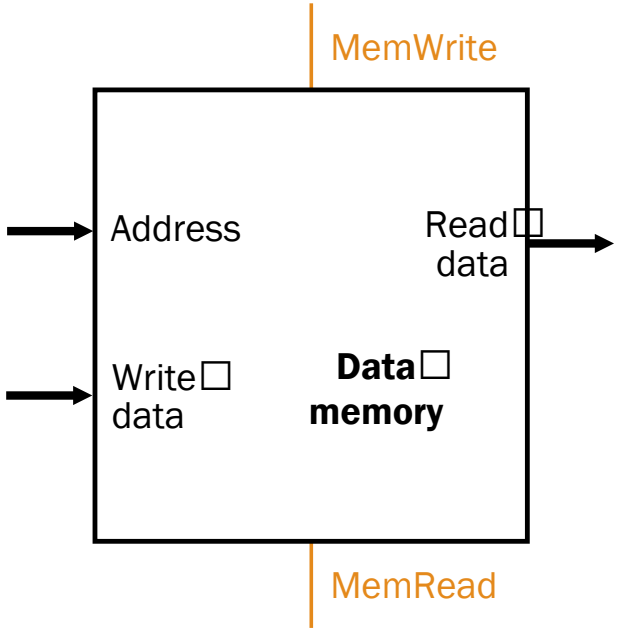
- **add**
- **sub**
- **slt**



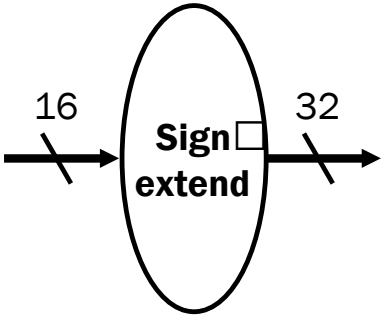
Suporte para instruções lw/sw

- **lw $t1, D($t2)**
- **sw $t1, D($t2)**

D: deslocamento (16 bits)



a. Data memory unit



b. Sign-extension unit

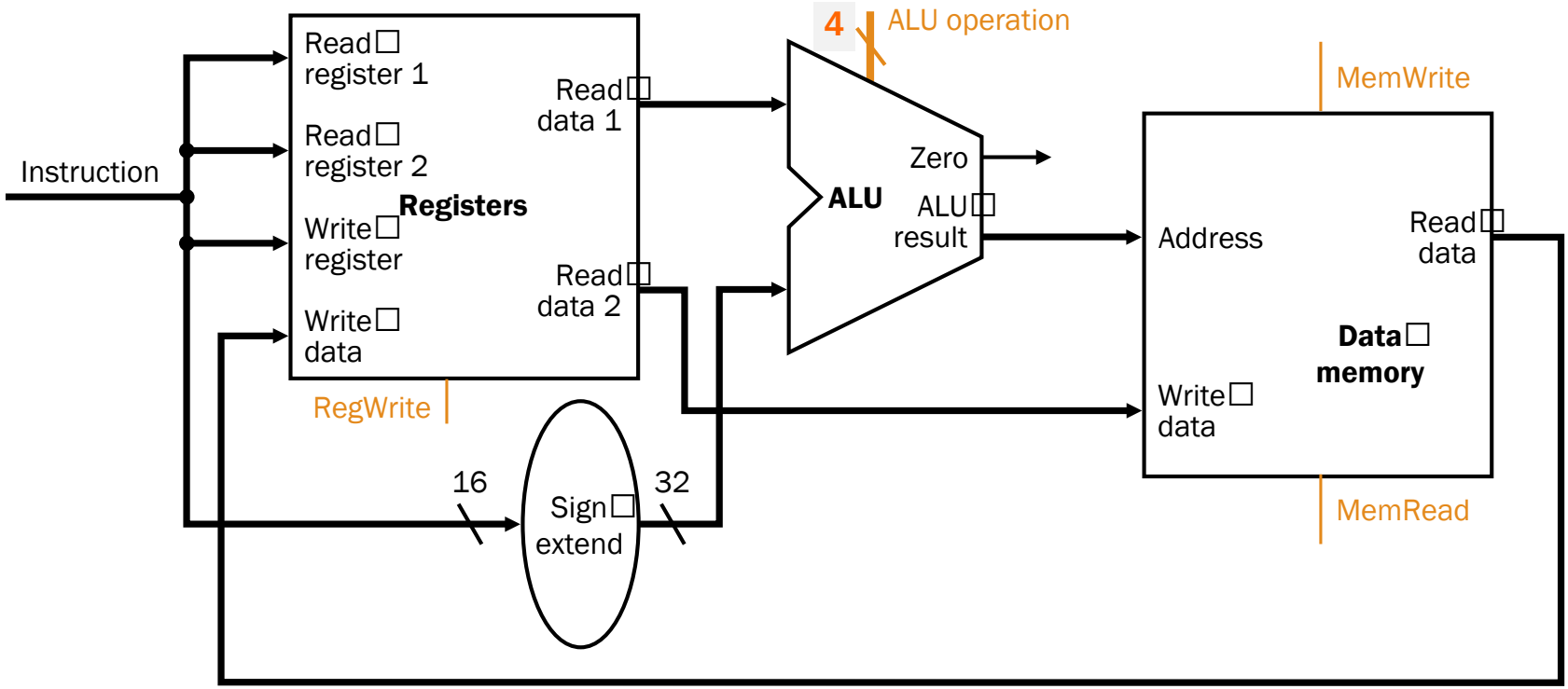
Banco de registradores

+

ALU

Suporte para instruções lw/sw

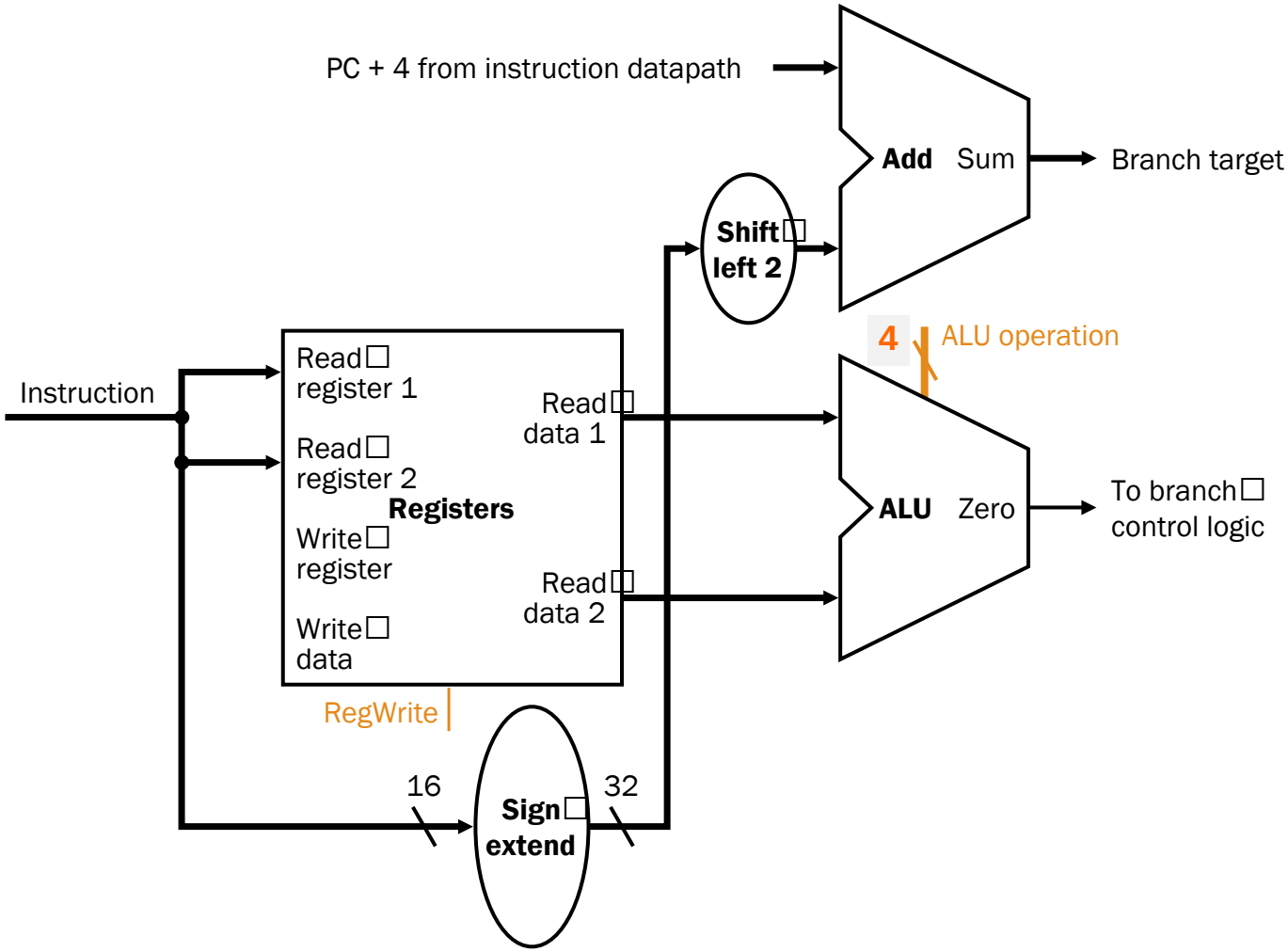
- **lw $t1, D($t2)**
- **sw $t1, D($t2)**



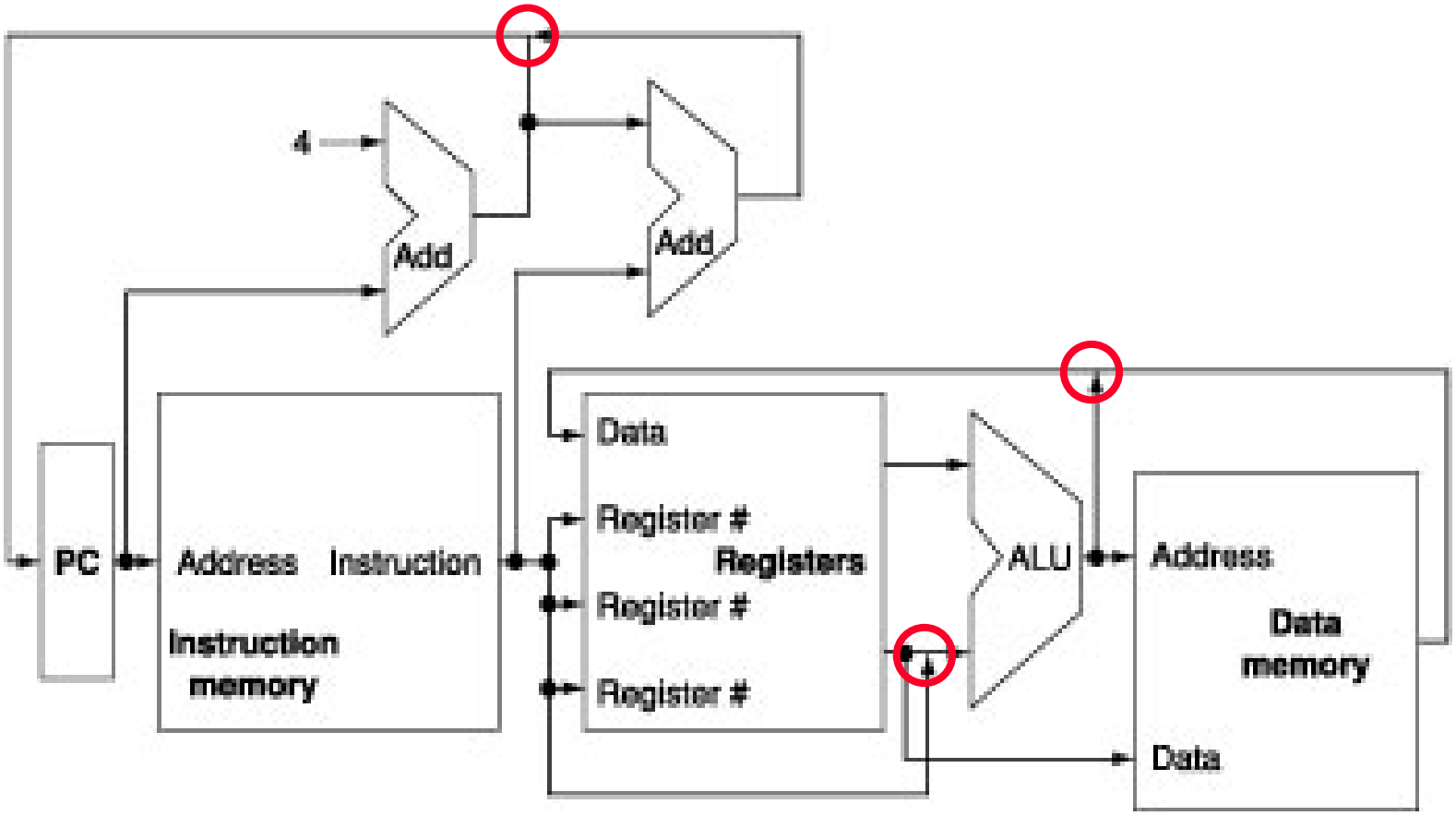
Suporte para instrução beq

- **beq $t1, $t2, D**
- **Detalhes na interpretação do deslocamento**
  - **Deslocamento a partir de PC+4**
  - **O valor de D sofre deslocamento de 2 bits para a esquerda**
- **Operandos iguais**
  - **Desvio realizado: PC ← novo PC**
- **Operandos diferentes**
  - **Desvio não realizado: PC já incrementado**

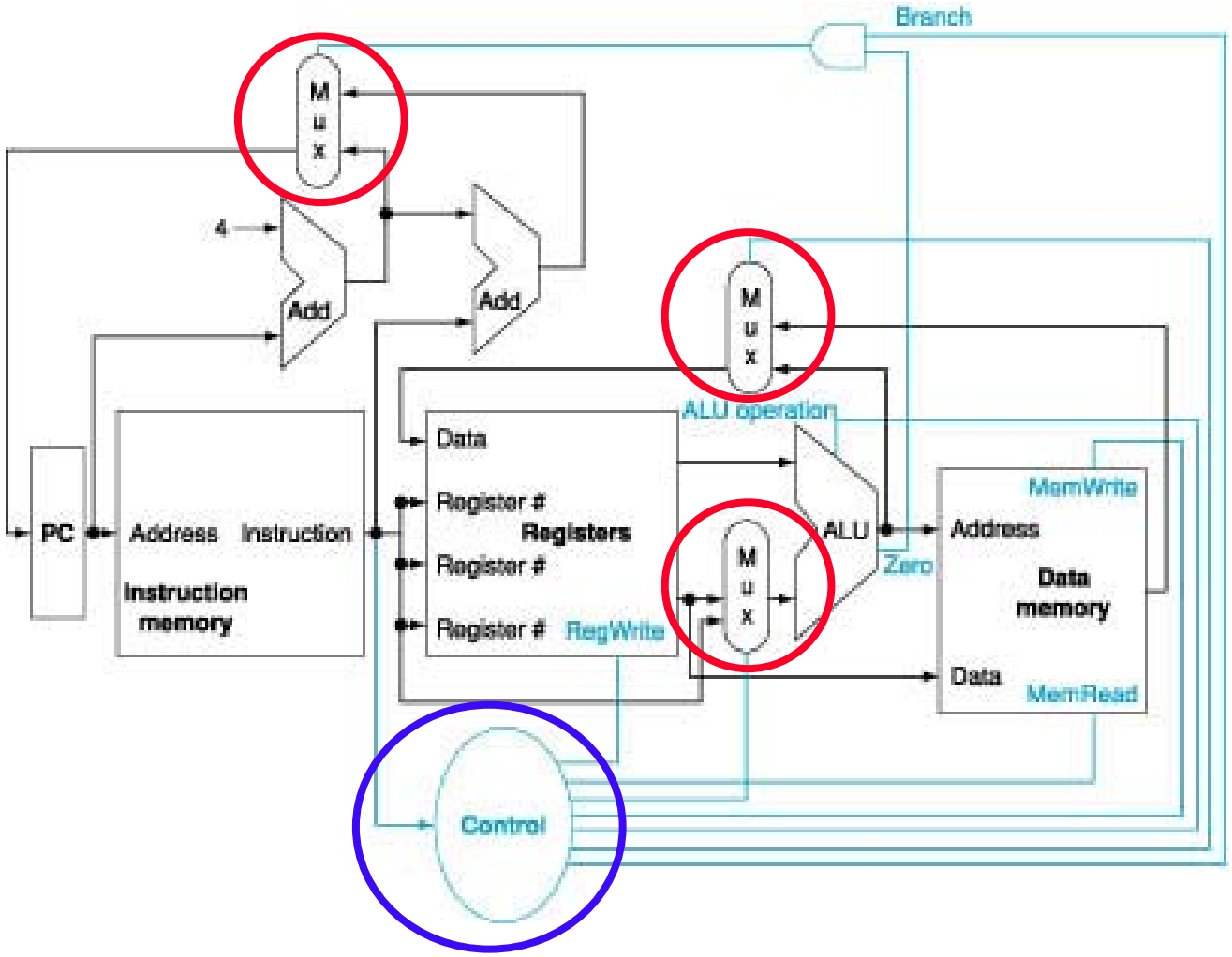
Suporte para instrução beq



Datapath: uma visão abstrata

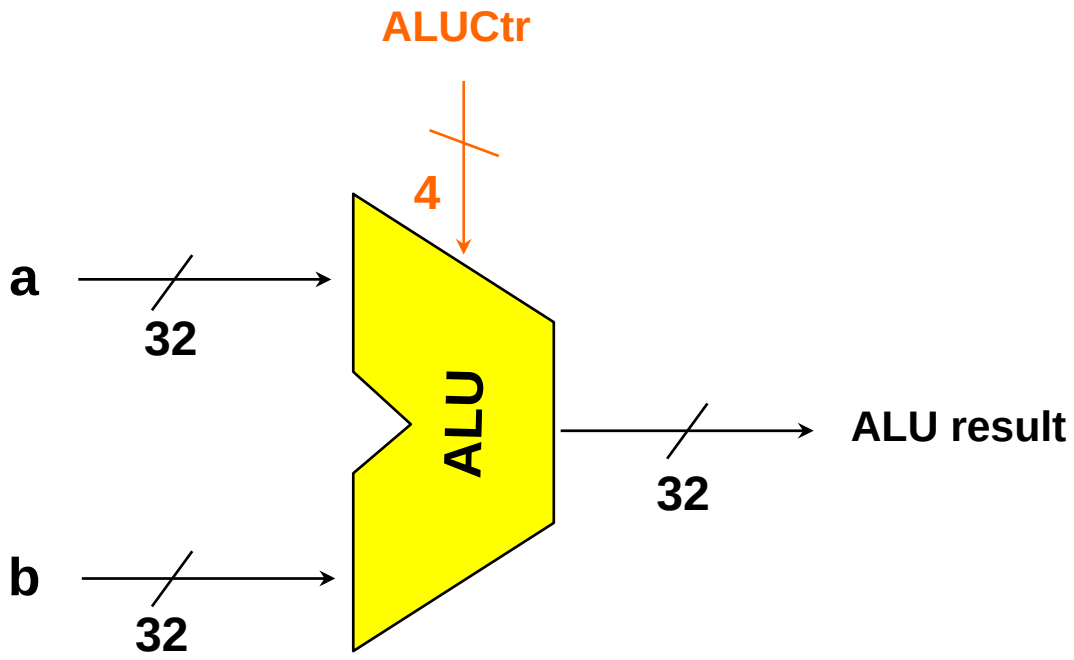


Incluindo muxes e controlador



ALU

- **Projeto em nível lógico**
  - **Diagrama esquemático no Apêndice C**



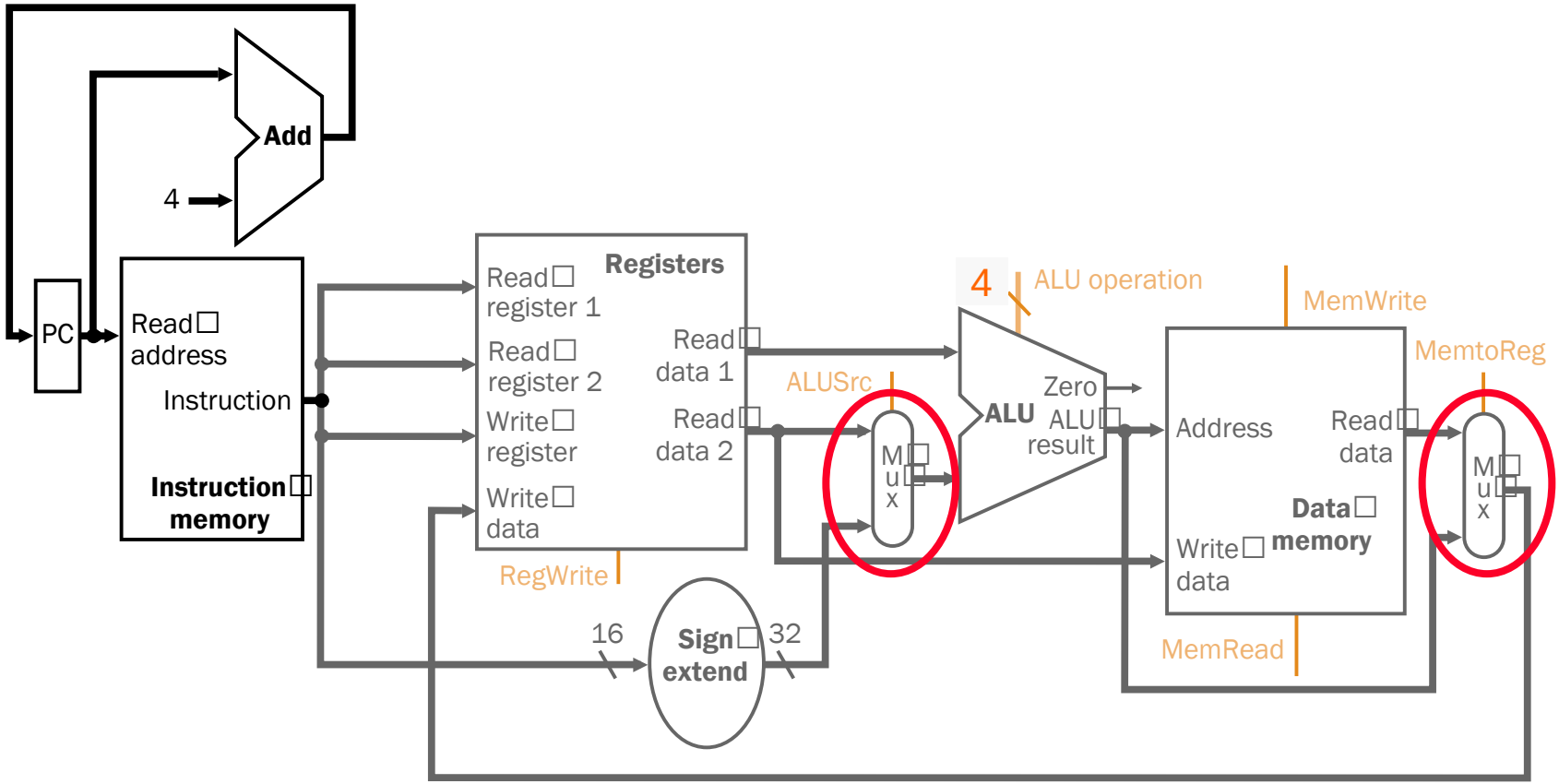
ALU

- **Projeto em nível lógico**
  - **Diagrama esquemático no Apêndice C**

```
0000 = and
0001 = or
0010 = add
0110 = subtract
0111 = slt
1100 = nor
```

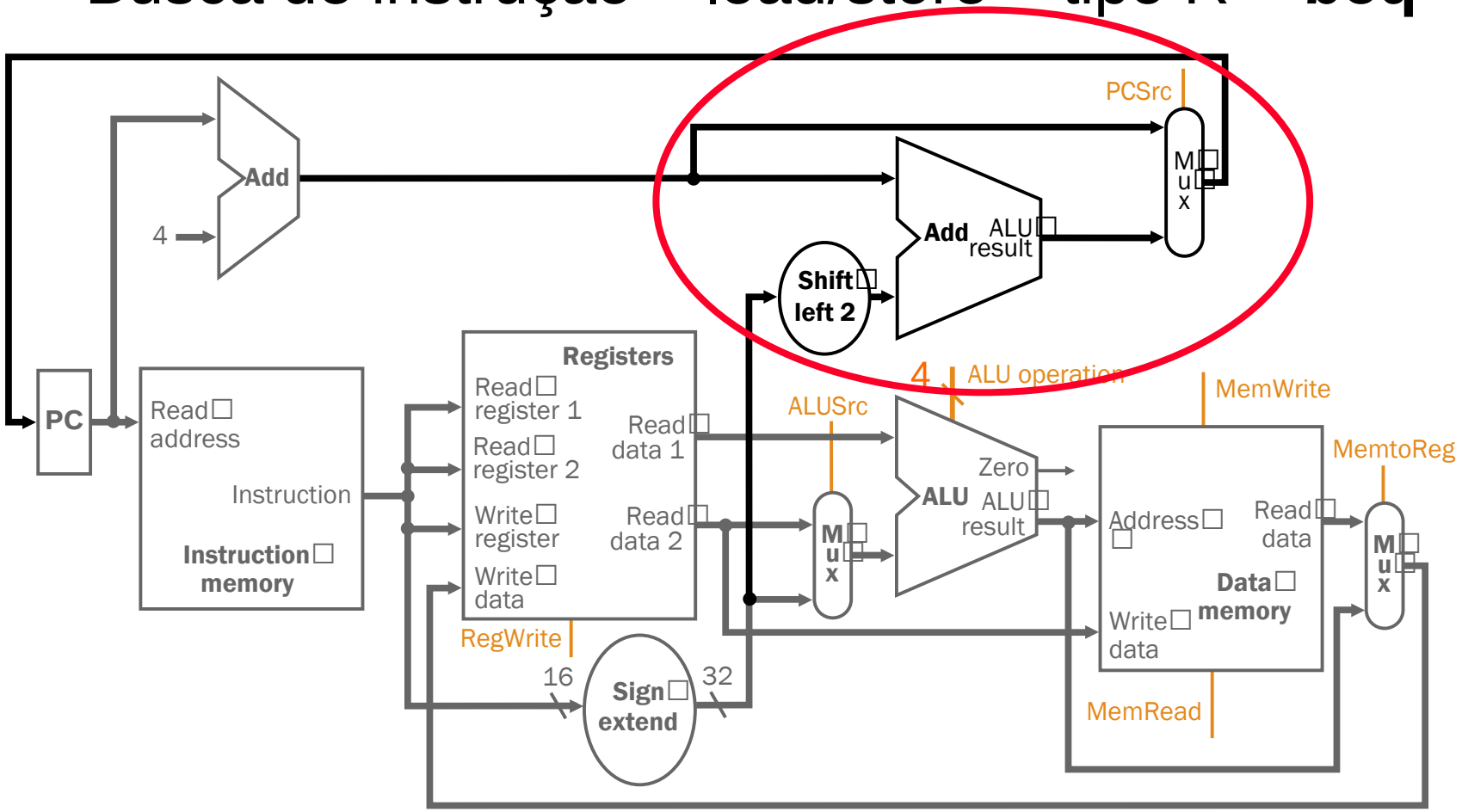
Compartilhamento de HW

- **Busca de instrução +** load/store **+** tipo R

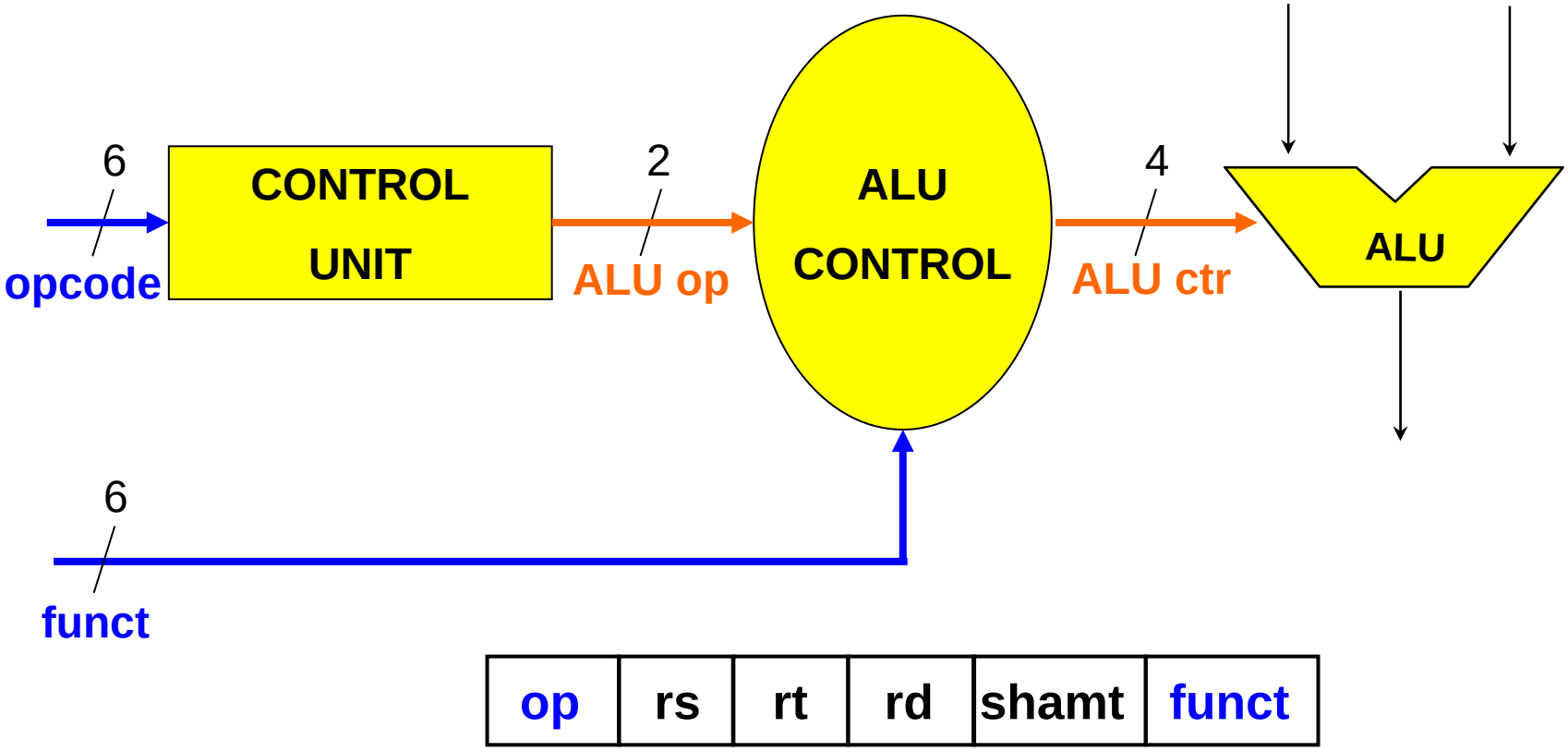


Incluindo o suporte para beq

- Busca de instrução **+** load/store **+** tipo R + **beq**



O controle da ALU



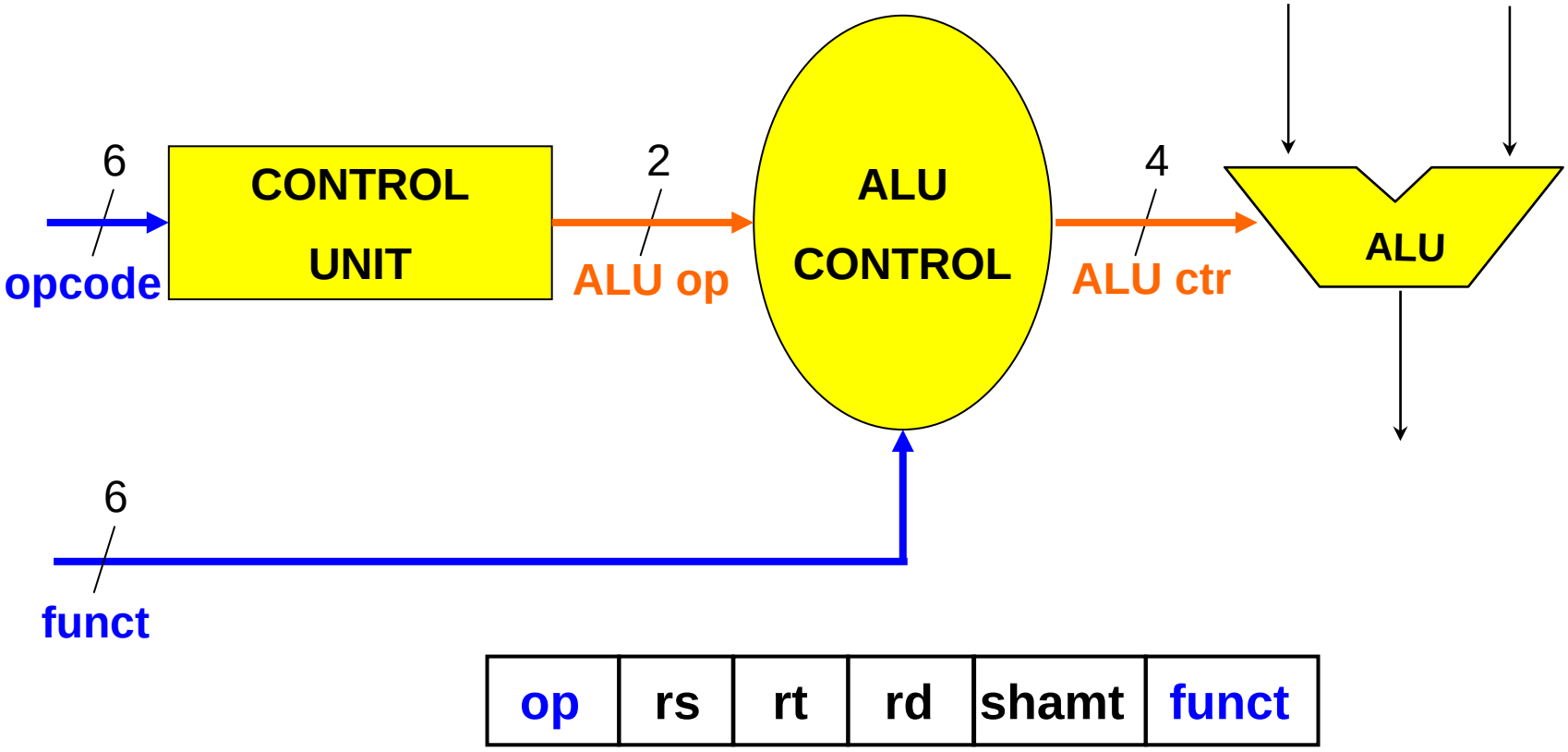
O controle da ALU

Opcode		Operação	Funct	Ação ALU	ALU ctr
100011		lw	xxxxxx	soma	0010
101011		sw	xxxxxx	soma	0010
000100		beq	xxxxxx	subtr	0110
000000	Tipo R	add	100000	soma	0010
000000		sub	100010	subtr	0110
000000		and	100100	and	0000
000000		or	100101	or	0001
000000		slt	101010	slt	0111

O controle da ALU

Opcode	ALUop	Operação	Funct	Ação ALU	ALU ctr
100011	00	lw	xxxxxx	soma	0010
101011	00	sw	xxxxxx	soma	0010
000100	01	beq	xxxxxx	subtr	0110
000000	10	add	100000	soma	0010
000000	10	sub	100010	subtr	0110
000000	10	and	100100	and	0000
000000	10	or	100101	or	0001
000000	10	slt	101010	slt	0111

O controle da ALU



Incluindo o controle da ALU

- **Campos do IR + controle ALU + mux p/ reg. destino**