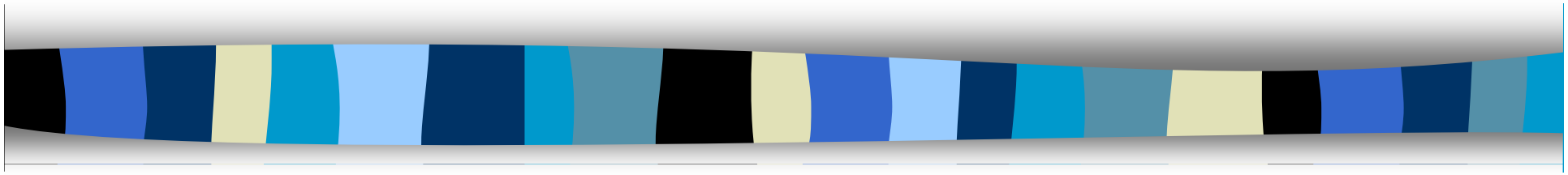
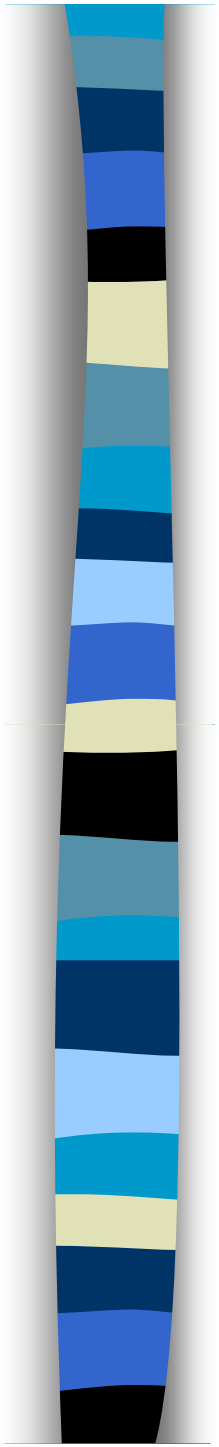
“Arquitetura e Organização de Computadores I – Aula_05 – Processador: Caminho de Dados e Controle”



Prof. Dr. Emerson Carlos Pedrino
DC/UFSCar
São Carlos

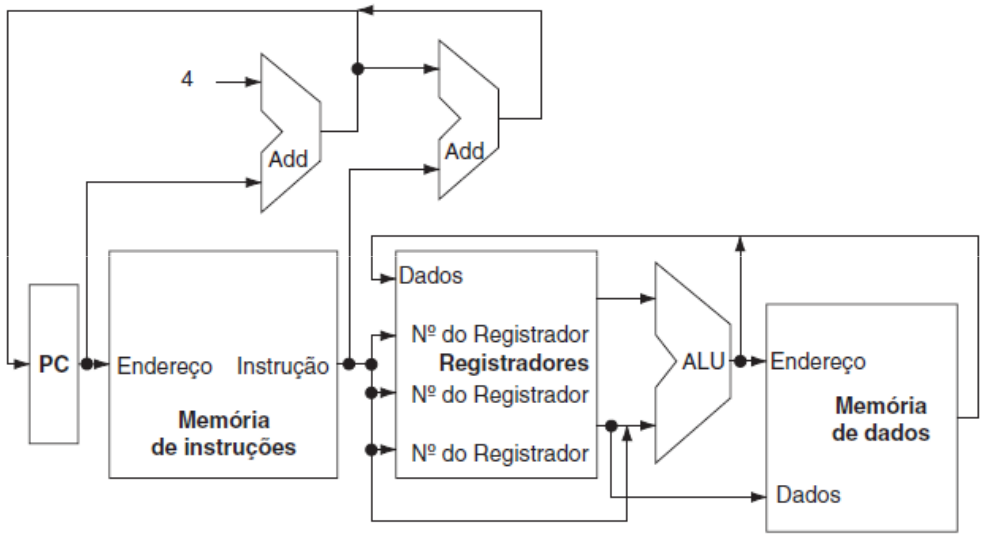


O Processador: Caminho de Dados e Controle

- Estamos prontos para ver uma implementação do MIPS
- Simplificada para conter apenas:
  - instruções de referência à memória: `lw`, `sw`
  - instruções lógicas e aritméticas: `add`, `sub`, `and`, `or`, `slt`
  - instruções de fluxo de controle: `beq`, `j`
- Implementação genérica:
  - use o contador de programa (PC) para fornecer endereço de instrução
  - obtenha a instrução da memória
  - leia os registradores
  - use a instrução para decidir exatamente o que fazer
- Todas as instruções usam a ALU após lerem os registradores
  Por quê? Referência à memória? Aritmética? Fluxo de controle?

Mais Detalhes de Implementação

- **Visão abstrata/simplificada:**



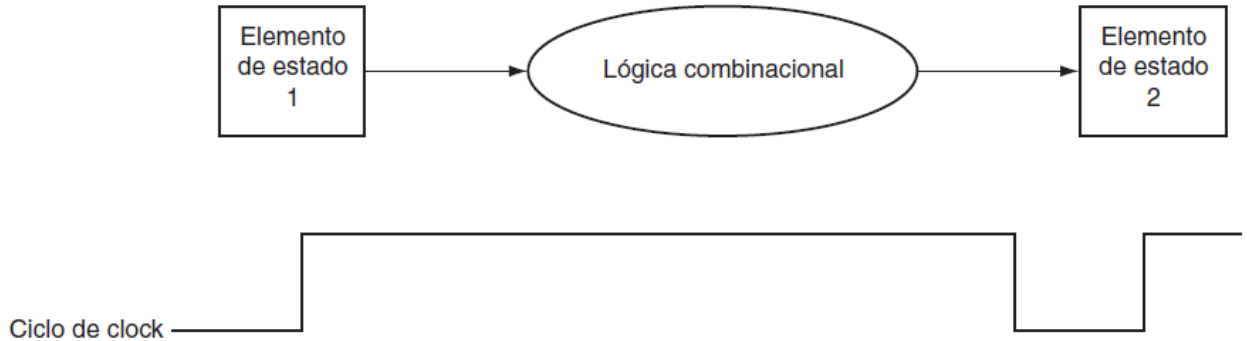
Mais Detalhes de Implementação

Dois tipos de unidades funcionais:

- elementos que operam nos valores de dados (combinacionais)
- elementos que contêm estado (seqüenciais)

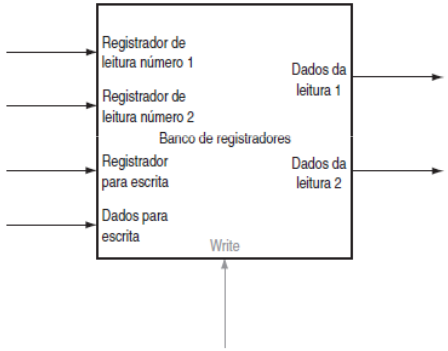
Nossa Implementação

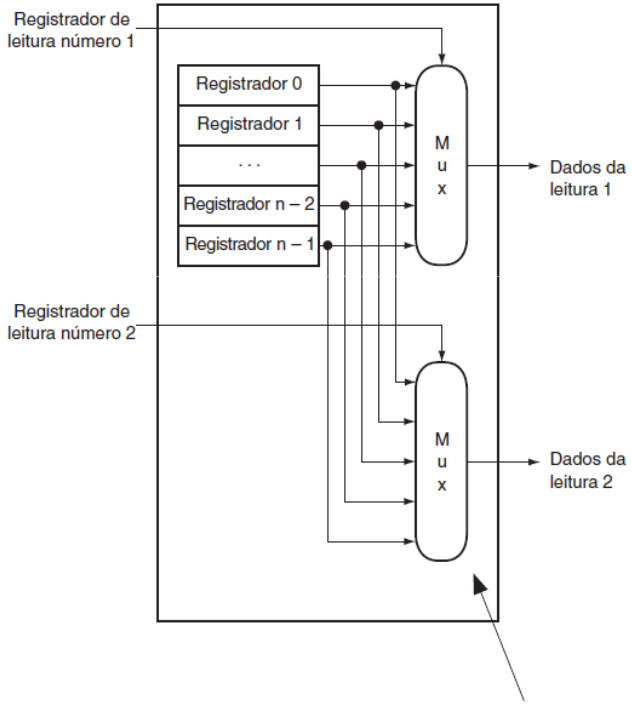
- **Uma metodologia de acionamento por transição**
- **Execução típica:**
  - **ler conteúdo de alguns elementos de estado,**
  - **enviar valores através de alguma lógica combinacional**
  - **escrever os resultados em um ou mais elementos de estado**



Arquivo de Registrador

- **Construído usando flip-flops D**

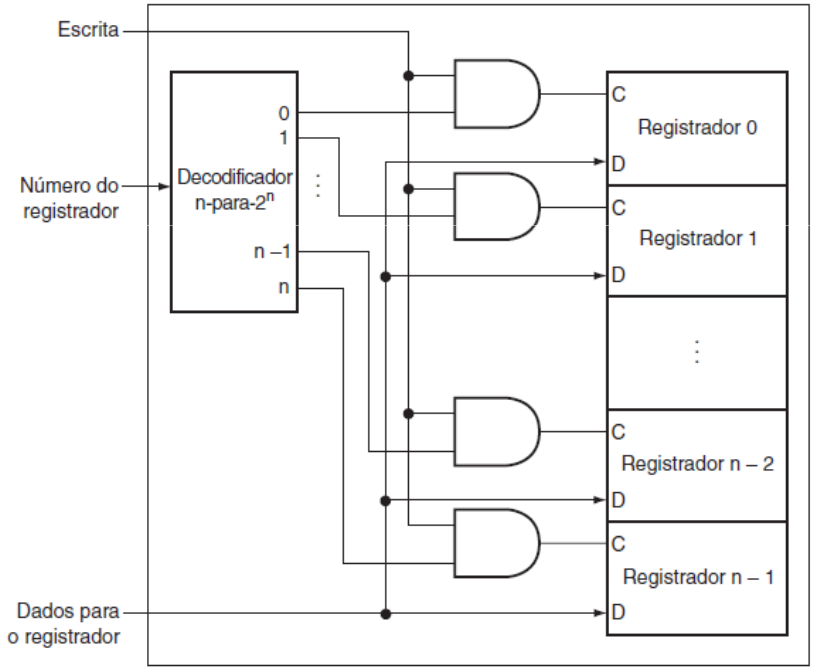




Você entende? Qual é o "Mux" acima?

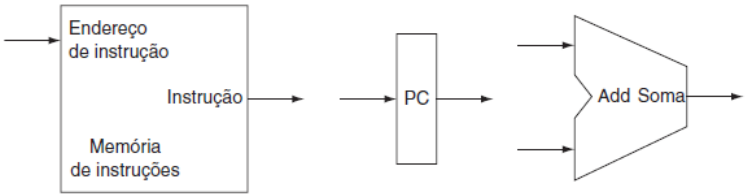
Arquivo de Registrador

- **Nota: ainda usamos o clock real para determinar quando escrever**

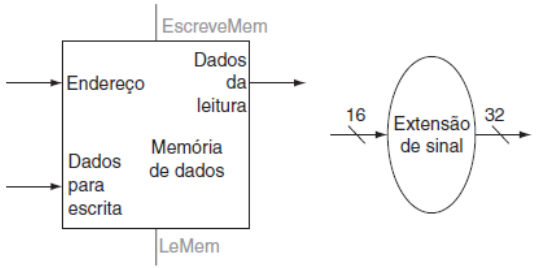


Implementação Simples

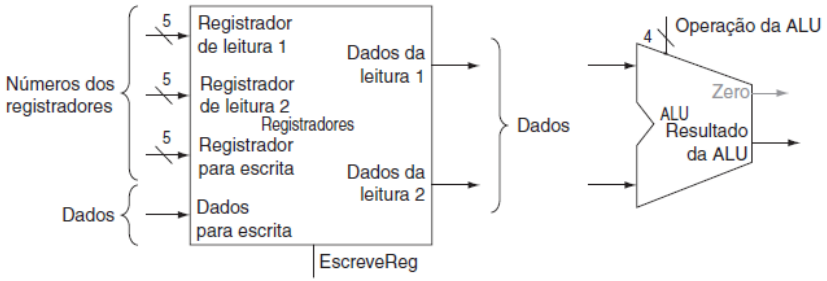
- **Inclui as unidades funcionais de que precisamos para cada instrução**



a. Memória de instruções b. Contador de programa c. Somador



a. Unidade de memória de dados b. Unidade de extensão de sinal

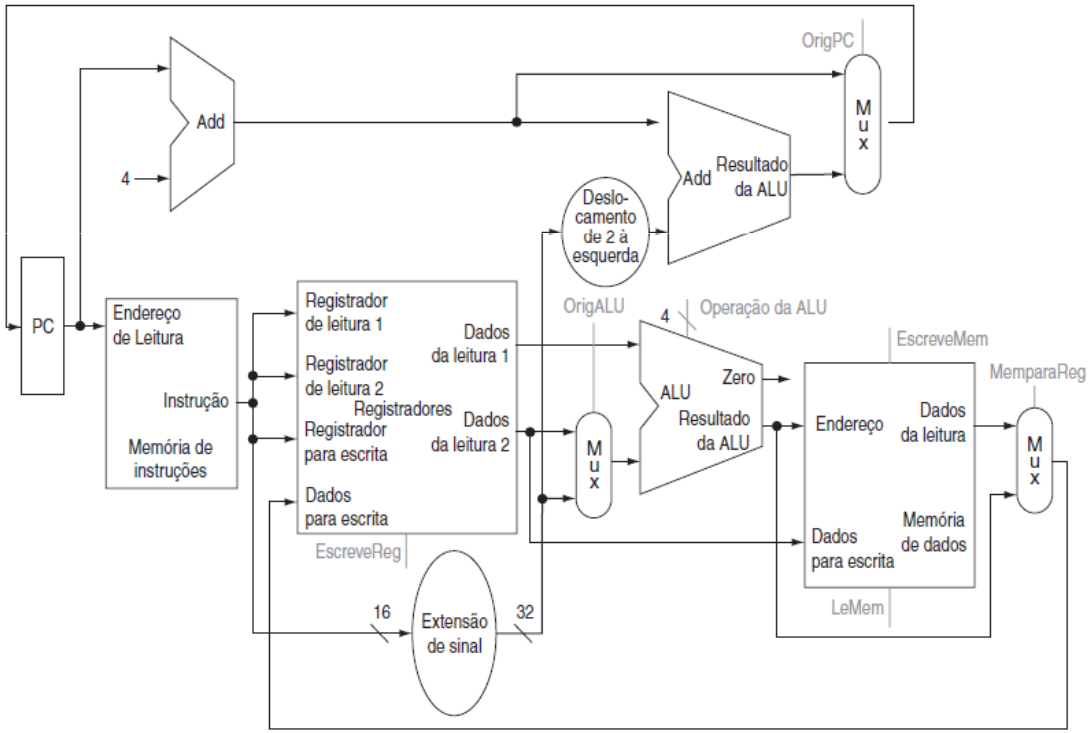


a. Registradores b. ALU

Por que precisamos disso?

Construindo o Caminho de Dados

- **Use multiplexadores para uni-los**



O Controle da ALU

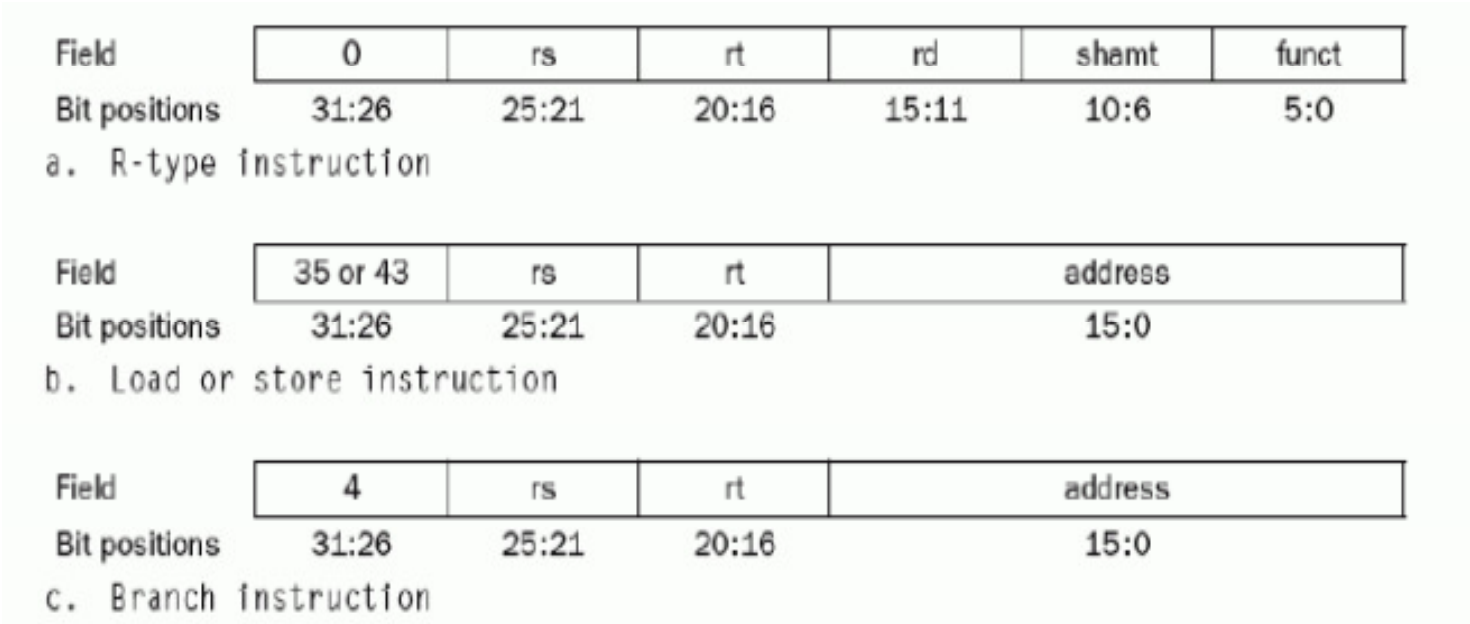
ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

OpALU+funct -> Entrada da Unidade de Controle

Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	and	0000
R-type	10	OR	100101	or	0001
R-type	10	set on less than	101010	set on less than	0111

Projetando a Unidade de Controle Principal

- Classes de instrução



Field	0	rs	rt	rd	shamt	funct
Bit positions	31:26	25:21	20:16	15:11	10:6	5:0

a. R-type instruction

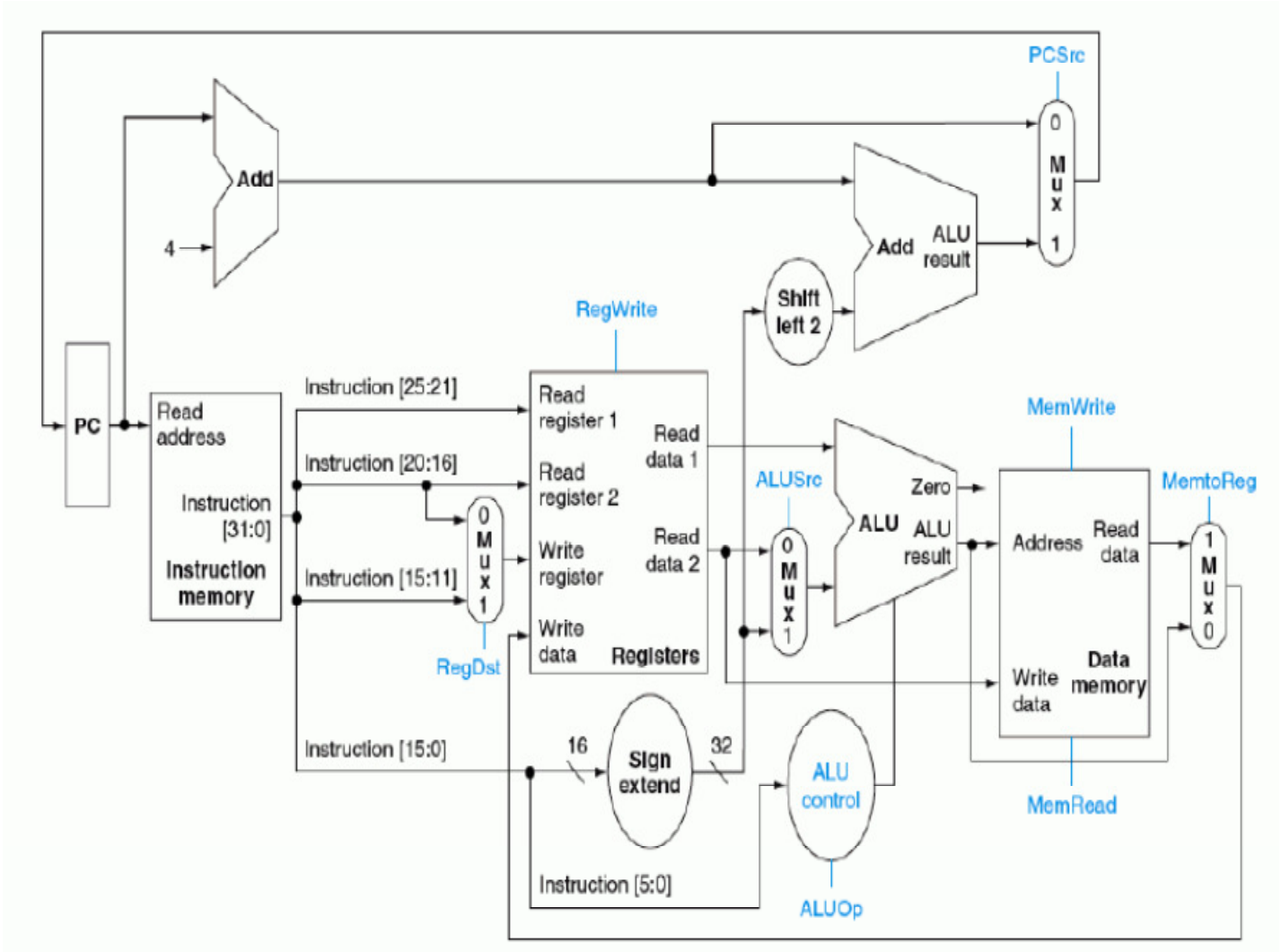
Field	35 or 43	rs	rt	address
Bit positions	31:26	25:21	20:16	15:0

b. Load or store instruction

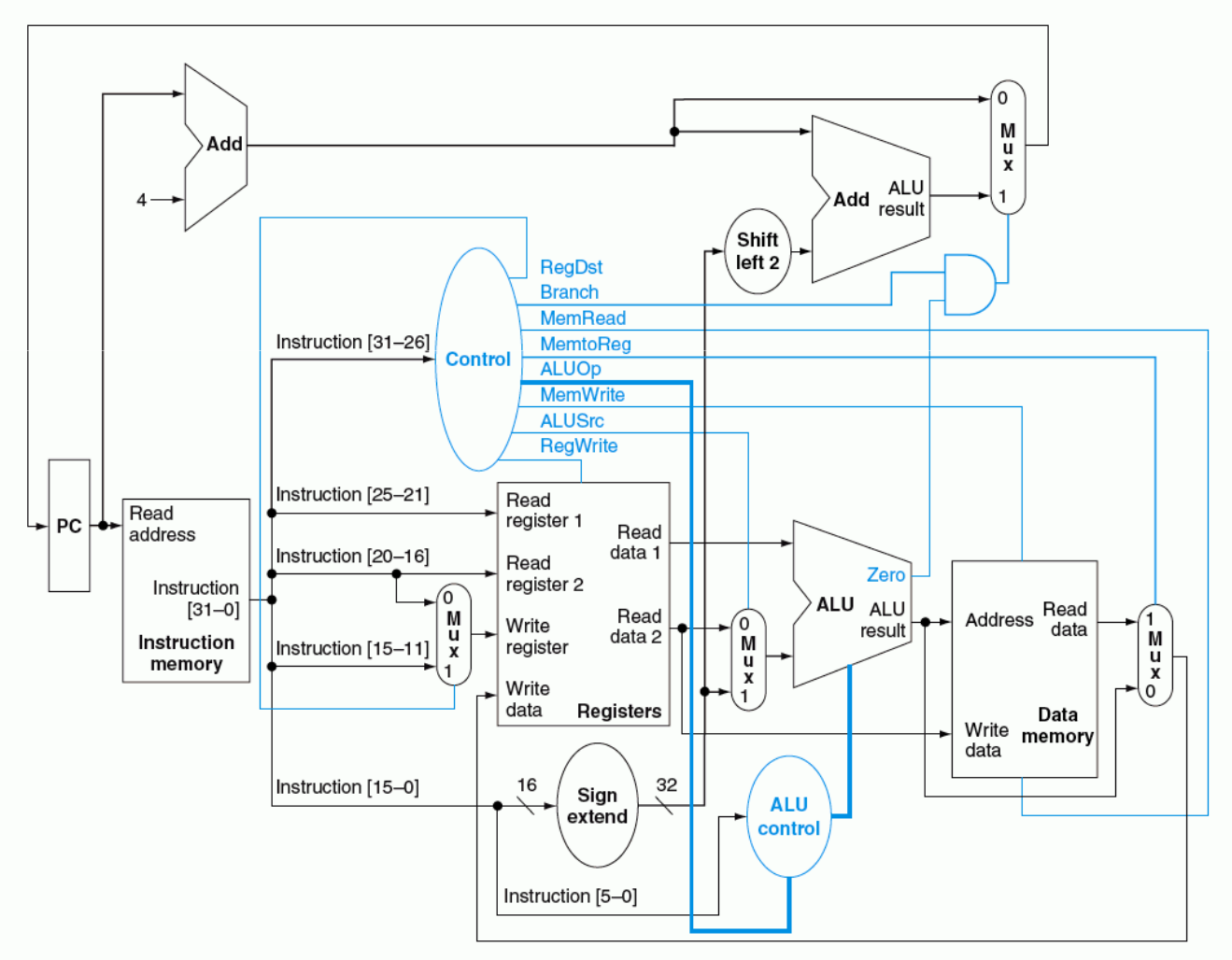
Field	4	rs	rt	address
Bit positions	31:26	25:21	20:16	15:0

c. Branch instruction

Caminho de Dados



O Caminho de Dados Simples com a Unidade de Controle



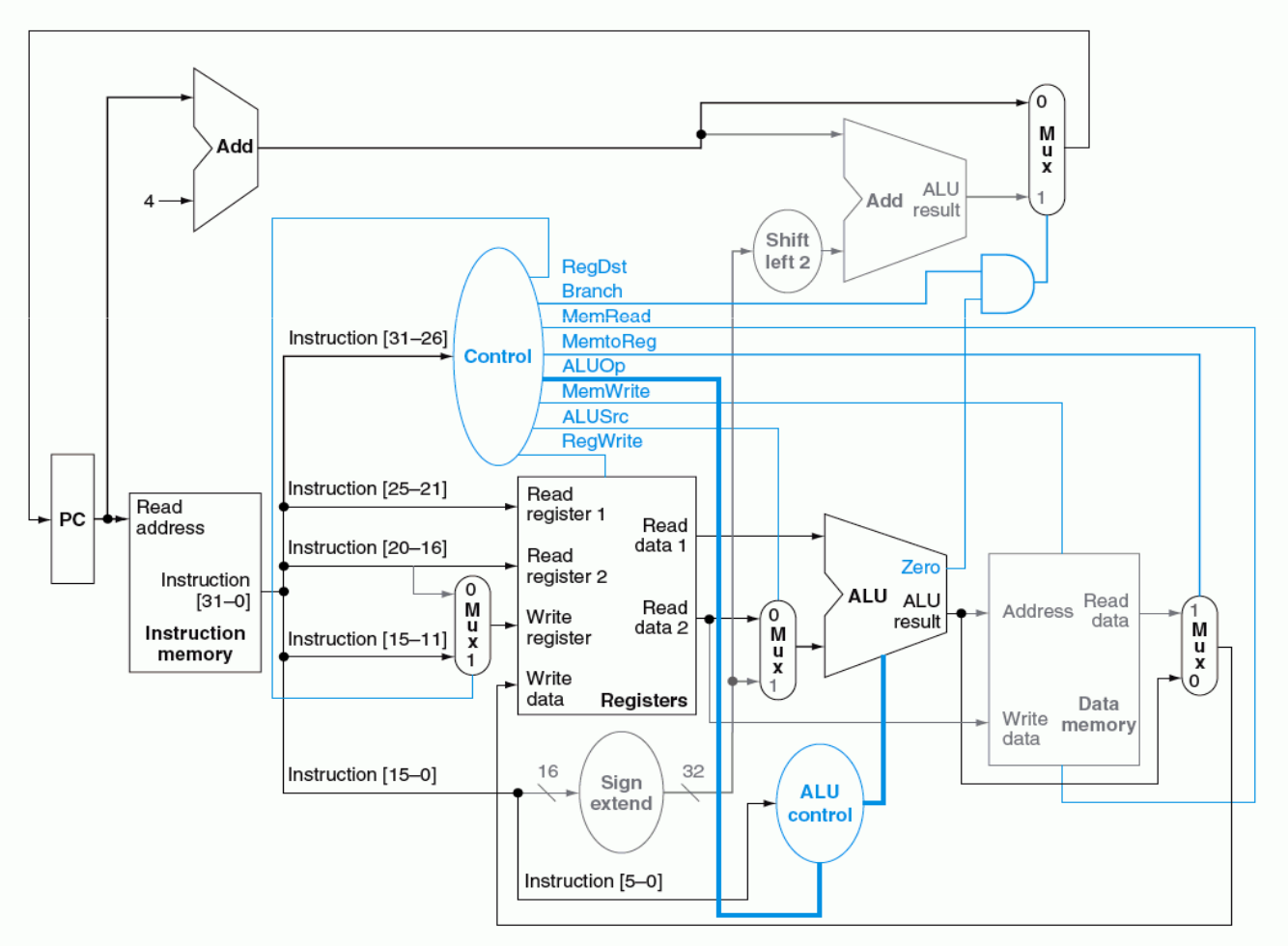
Definição das Linhas de Controle – Determinada pelo *opcode*

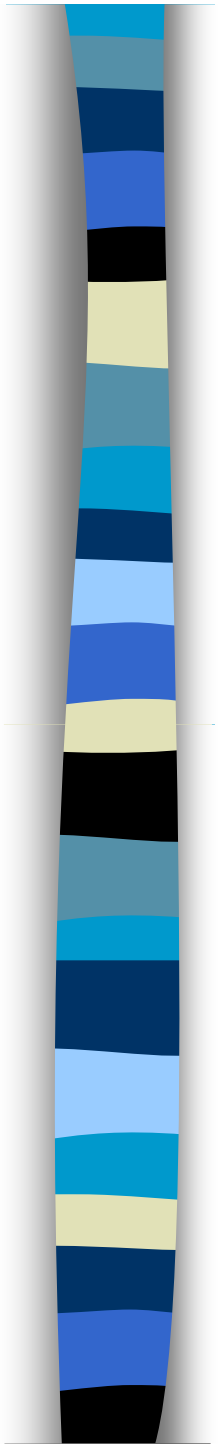
Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

Caminho de Dados para uma instrução do Tipo R

- Exemplo: add $t1,$t2,$t3 (um único ciclo de clk):
  - 1. A instrução é buscada e o PC é incrementado.
  - 2. Dois registradores são lidos do banco de registradores e a UC principal calcula a definição das linhas de controle.
  - ALU: opera nos dados lidos usando o código de função.
  - O resultado da ALU é escrito no banco de registradores através dos bits 15:11.

Caminho de Dados para uma instrução do Tipo R





Exercício*

- Ilustrar a execução de um *load word*.
- Ilustrar a execução de um *branch-on-equal.*

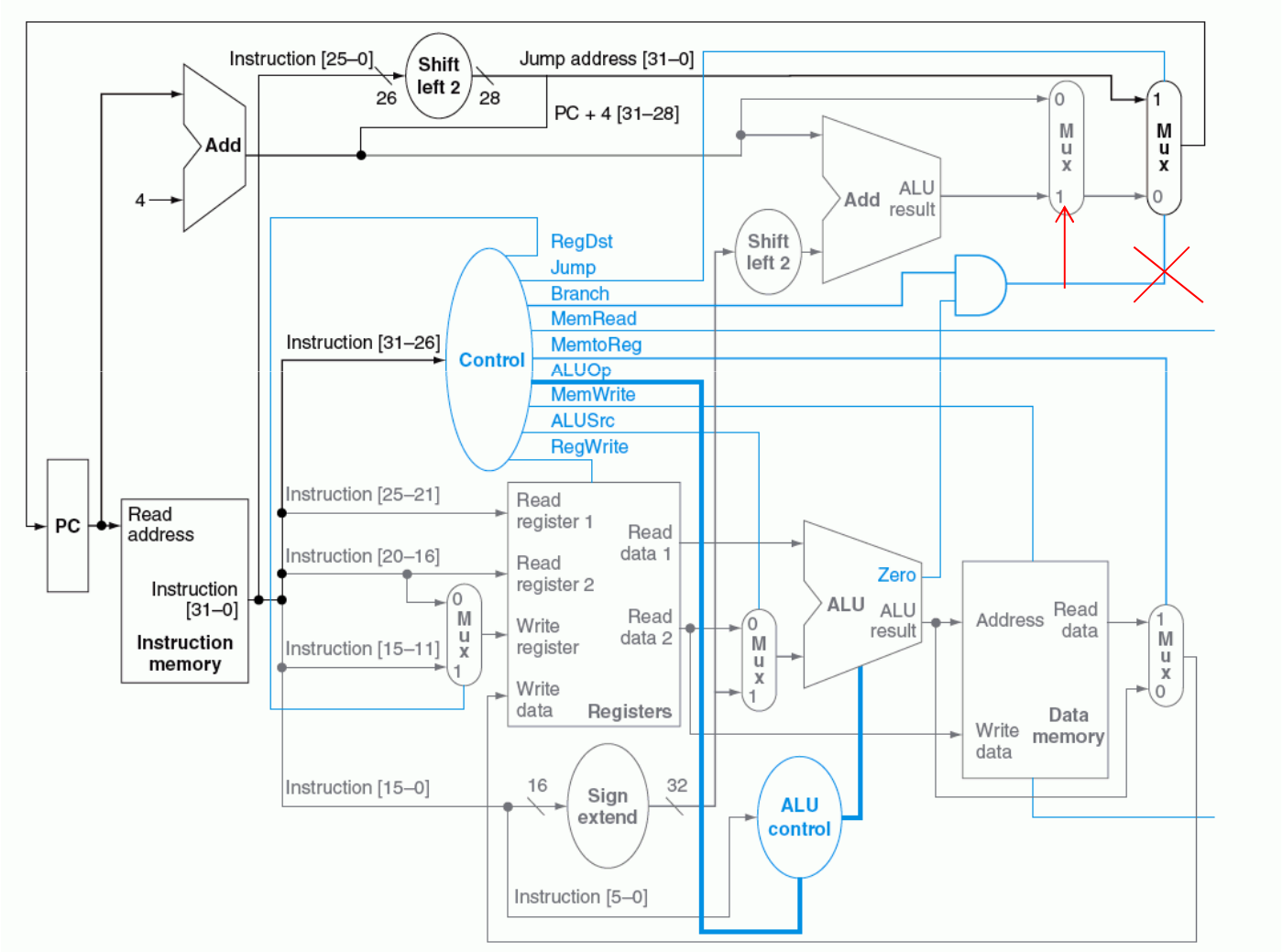
Função de Controle – Implementação Simples de 1 ciclo

Input or output	Signal name	R-format	lw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

Exercício*

- Implemente o decodificador simplificado para a função de controle de ciclo único vista anteriormente.

Instrução JUMP



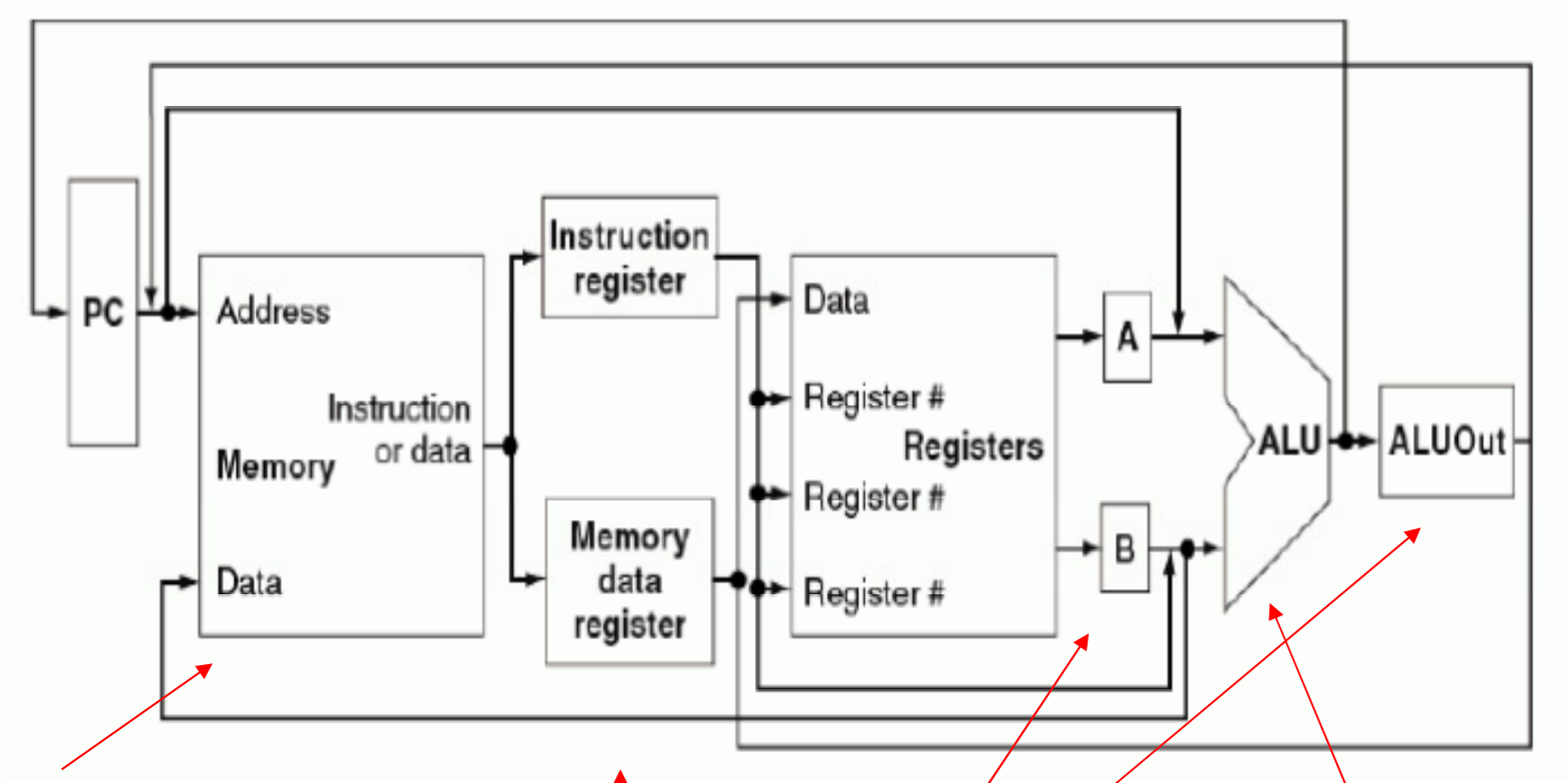
Observações

- Projeto de ciclo único -> ineficiente.
- CPI -> 1.
- Ciclo de clk -> determinado pelo caminho mais longo possível na máquina (instrução *load*, usa 5 unidades funcionais: mem_instr, bc_regs, ULA, mem_dat, bc_regs).

Implementação Multiciclo

- Cada etapa na execução levará um ciclo de clk.
- Uma unidade funcional pode ser usada mais de uma vez por instrução em diferentes ciclos de clk.
- Redução da quantidade de *hardware* necessária.

Visão de Alto Nível do Caminho de Dados Multiciclo

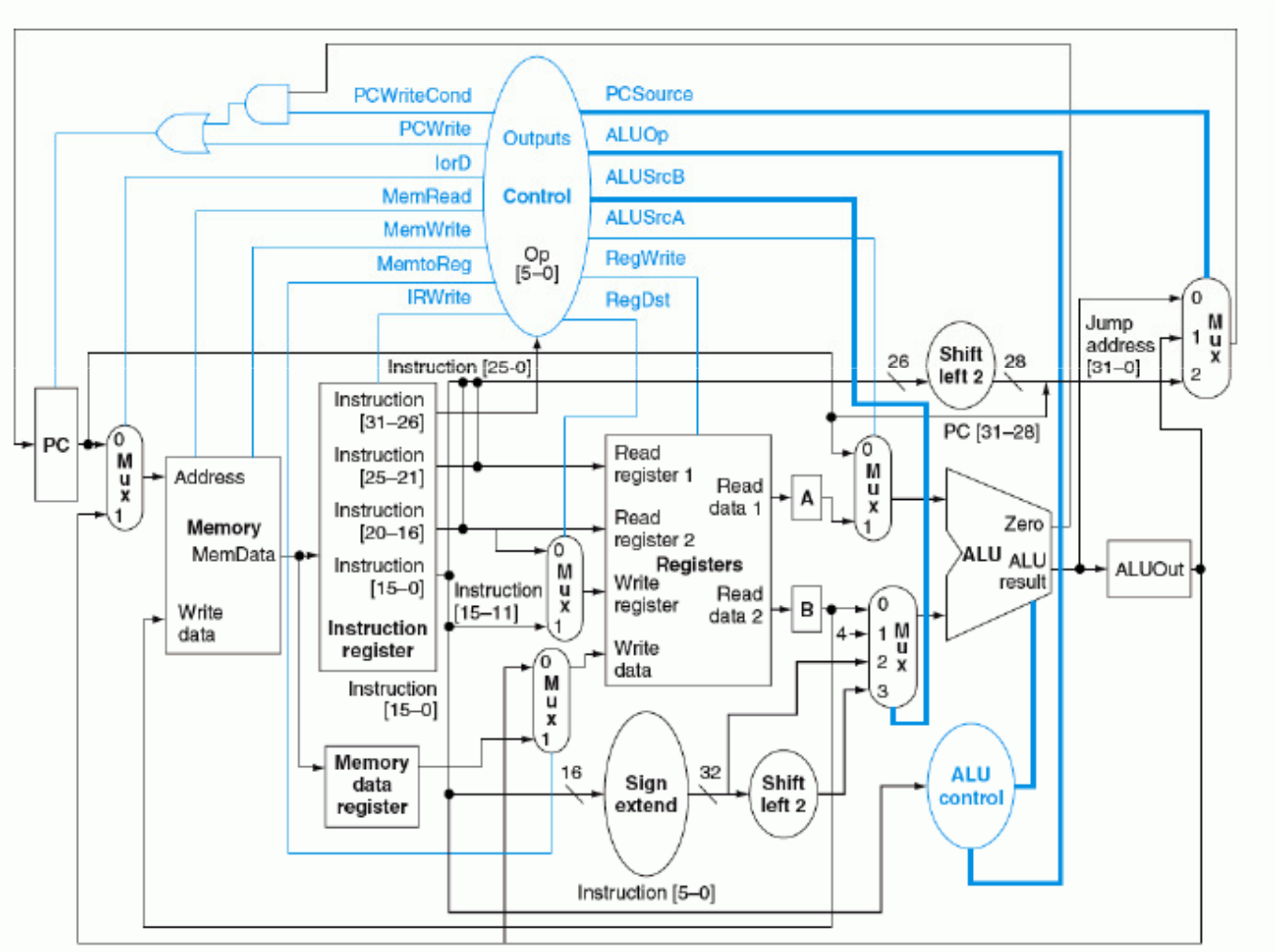


memória única

novos regs

ALU única

Caminho de Dados Completo



Ações dos Sinais de Controle de 1 *bit*

Signal name	Effect when deasserted	Effect when asserted
RegDst	The register file destination number for the Write register comes from the rt field.	The register file destination number for the Write register comes from the rd field.
RegWrite	None.	The general-purpose register selected by the Write register number is written with the value of the Write data input.
ALUSrcA	The first ALU operand is the PC.	The first ALU operand comes from the A register.
MemRead	None.	Content of memory at the location specified by the Address input is put on Memory data output.
MemWrite	None.	Memory contents at the location specified by the Address input is replaced by value on Write data input.
MemtoReg	The value fed to the register file Write data input comes from ALUOut.	The value fed to the register file Write data input comes from the MDR.
IorD	The PC is used to supply the address to the memory unit.	ALUOut is used to supply the address to the memory unit.
IRWrite	None.	The output of the memory is written into the IR.
PCWrite	None.	The PC is written; the source is controlled by PCSource.
PCWriteCond	None.	The PC is written if the Zero output from the ALU is also active.

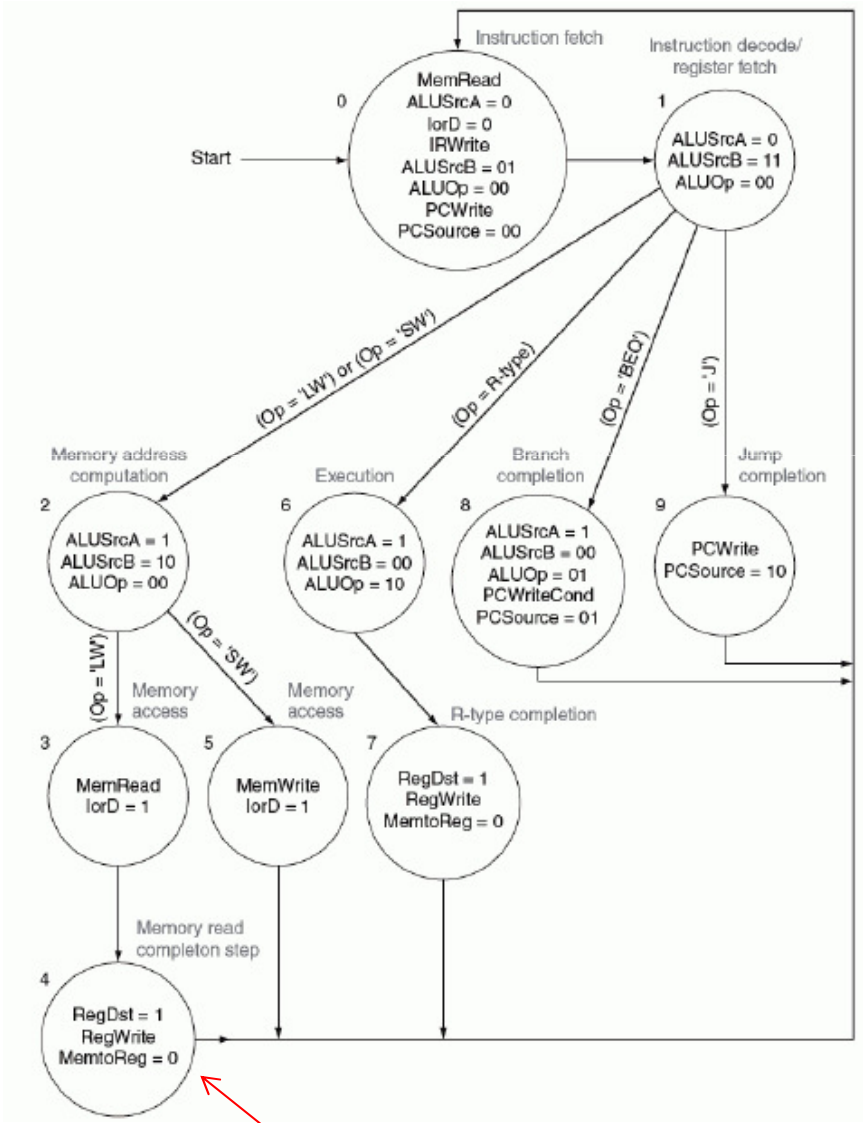
Ações dos Sinais de Controle de 2 *bits*

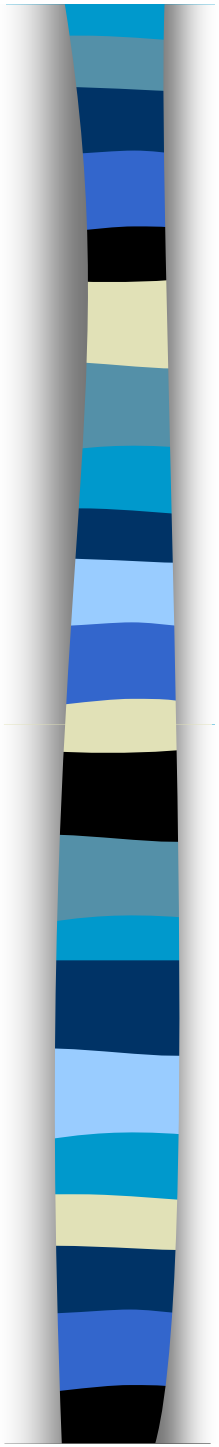
Signal name	Value (binary)	Effect
ALUOp	00	The ALU performs an add operation.
	01	The ALU performs a subtract operation.
	10	The funct field of the instruction determines the ALU operation.
ALUSrcB	00	The second input to the ALU comes from the B register.
	01	The second input to the ALU is the constant 4.
	10	The second input to the ALU is the sign-extended, lower 16 bits of the IR.
	11	The second input to the ALU is the sign-extended, lower 16 bits of the IR shifted left 2 bits.
PCSource	00	Output of the ALU (PC + 4) is sent to the PC for writing.
	01	The contents of ALUOut (the branch target address) are sent to the PC for writing.
	10	The jump target address (IR[25:0] shifted left 2 bits and concatenated with PC + 4[31:28]) is sent to the PC for writing.

Dividindo a Execução de Instrução em Ciclos de clk

Step name	Action for R-type instructions	Action for memory-reference instructions	Action for branches	Action for jumps
Instruction fetch	IR <= Memory[PC] PC <= PC + 4			
Instruction decode/register fetch	A <= Reg [IR[25:21]] B <= Reg [IR[20:16]] ALUOut <= PC + (sign-extend (IR[15:0]) << 2)			
Execution, address computation, branch/jump completion	ALUOut <= A op B	ALUOut <= A + sign-extend (IR[15:0])	if (A == B) PC <= ALUOut	PC <= {PC [31:28], (IR[25:0]),2'b00)}
Memory access or R-type completion	Reg [IR[15:11]] <= ALUOut	Load: MDR <= Memory[ALUOut] or Store: Memory [ALUOut] <= B		
Memory read completion		Load: Reg[IR[20:16]] <= MDR		

Controle da Máquina de Estados Finitos Completo para o Caminho de Dados Anterior

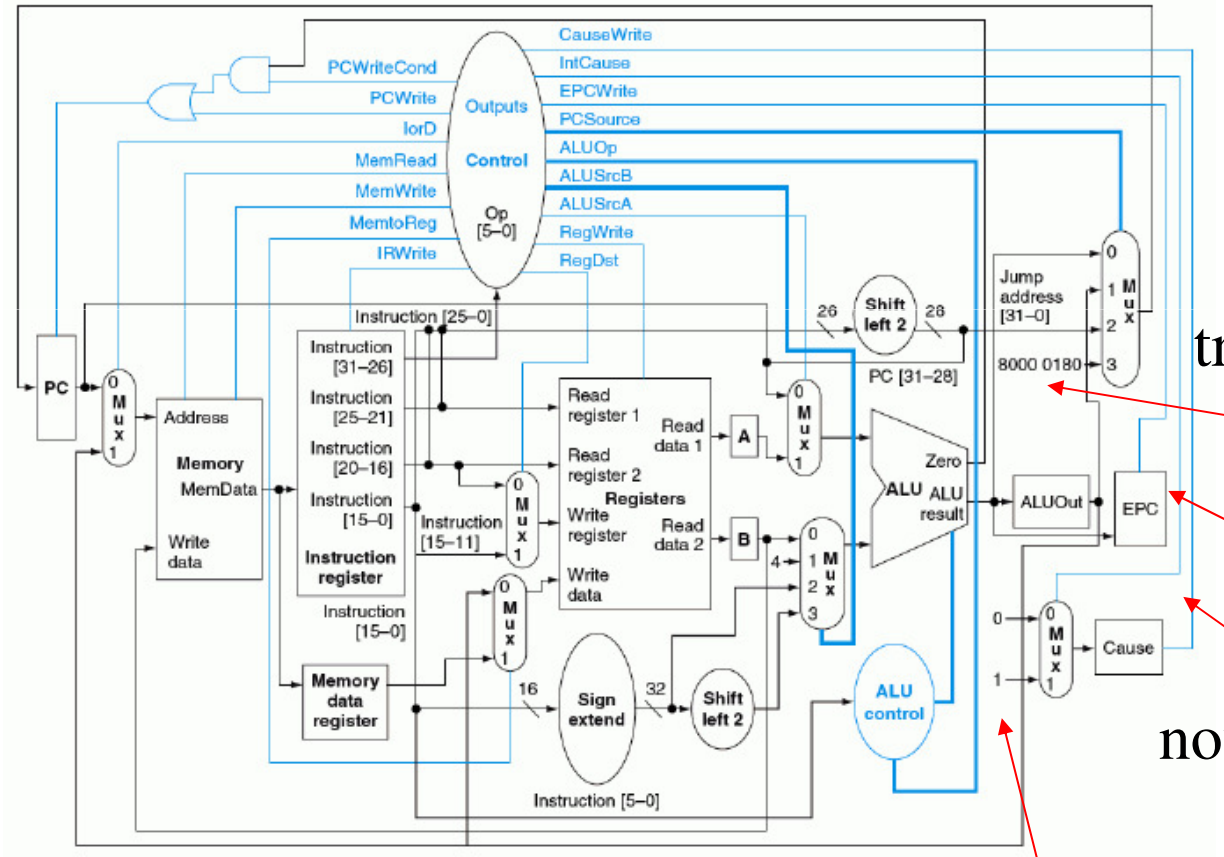




Exceções

- Também chamada de interrupção. Um evento não programado que interrompe a execução de um programa. Usada para detectar *Overflows*.

Caminho de Dados Multiciclo para Implementar Exceções



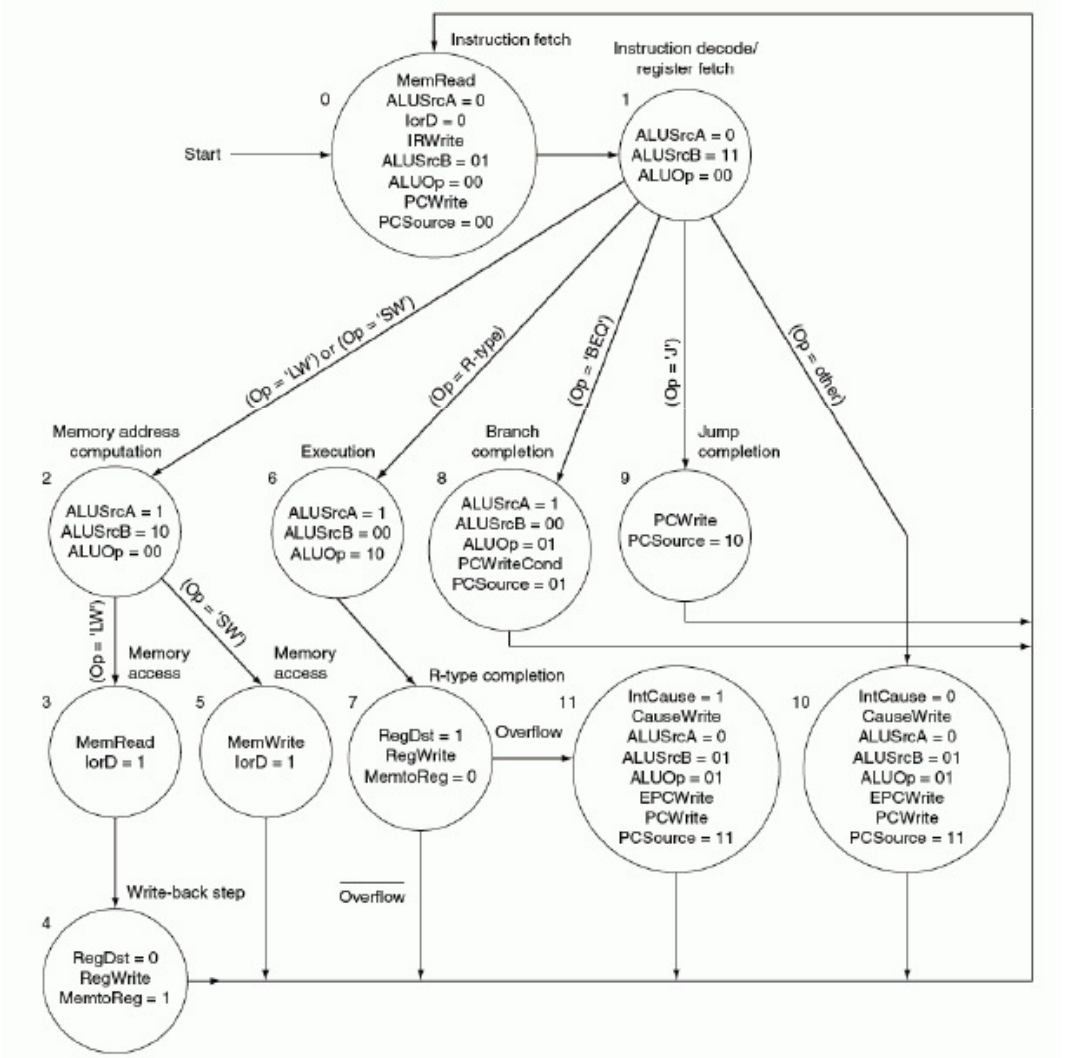
tratamento da exc.

PC-4

novos regs

0 – instr. Indef. 1- Ovf

Máquina de Estados Atualizada



Exercícios*

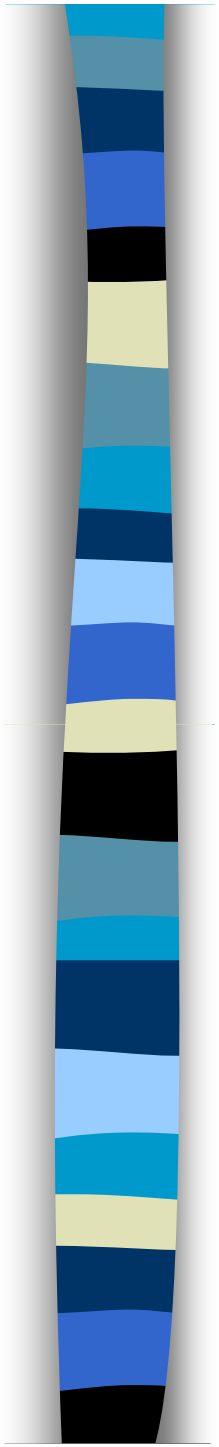
- 1. Mostre como seria o caminho de dados em operação para uma instrução *store* em uma máquina de ciclo único.

Exercícios*

- 2. Suponha que os tempos de operação para as principais unidades funcionais numa máquina de ciclo único sejam:
  - Unidades de memória: 200 ps.
  - ALU e somadores: 100 ps.
  - Banco de registradores: 50 ps.
- Considerando que os multiplexadores, a unidade de controle, os acessos do PC, a unidade de extensão de sinal e os fios têm atrasos desprezíveis, qual das seguintes implementações seria mais rápida e por quanto?
  - Uma implementação em que toda instrução opera em um ciclo de *clk* de uma duração fixa.
  - Uma implementação em que toda instrução é executada em 1 ciclo de *clk* usando um *clk* de duração variável, que, para cada instrução, tem apenas a duração necessária.
    - Para comparar o desempenho, considere o seguinte mix de instruções: 25% *loads*, 10% *stores*, 45% instruções da ALU, 15% desvios e 5% *jumps*.

Exercícios*

- 3. Mostre como seriam os sinais de controle da UC de uma máquina multiciclo para as etapas de: a. busca de instrução, b. decodificação de instrução e busca de registradores, c. execução, cálculo de endereço de memória e conclusão de desvio, d. etapa de acesso à memória ou conclusão de instrução do tipo R, e. etapa de conclusão de leitura de memória.



Exercícios*

- 4. Qual é o CPI de uma CPU multiciclo para o seguinte mix de instruções: 25% de *loads*, 10% de *stores*, 11% de *branches*, 2% de *jumps* e 52% de *ULA*.